AFFTC-TIH-96-02

AD-A286 920

AIR FORCE FLIGHT TEST CENTER TOLAND USERS' GUIDE

A F F T C

KENT STANDLEY
Aerospace Engineer

OCTOBER 1996

TECHNICAL INFORMATION HANDBOOK

97 3 19 503

AIR FORCE FLIGHT TEST CENTER
EDWARDS AIR FORCE BASE, CALIFORNIA
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

97-00467

A-1

This handbook, AFFTC-TIH-96-02, *Air Force Flight Test Center TOLAND Users' Guide*, was prepared on behalf of the 412 TW/TSF, Flight Dynamics Division, Edwards Air Force Base, California 93524-6843

Prepared by:

KENT STANDLEY
Aerospace Engineer

This handbook has been reviewed and is approved for publication 23 October 1996

FRANK S. BROWN
Acting Chief, Performance Strategic Flight

for ROBERT D. EVANS
Chief, Flight Dynamics Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Final

4. TITLE AND SUBTITLE
AFFTC TOLAND Users' Guide

5. FUNDING NUMBERS
JON 99800000
PEC 65807F

6. AUTHOR(S)
Standley, Kent

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Flight Test Center
412 Test Wing/TSFC
195 East Popson Avenue Bldg 2750
Edwards AFB CA 93524-6841

8. PERFORMING ORGANIZATION REPORT NUMBER
AFFTC-TIH-96-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
N/A

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(Maximum 200 words)*

This report describes a computer program that will model the takeoff and landing performance of aircraft. The program is written in Microsoft™ FORTRAN for execution on an IBM-compatible personal computer running with DOS 3.3 or better. Models of lift coefficient, drag coefficient, gross thrust, propulsive drag, and fuel flow within user-provided subroutines are required. The simulated aircraft can be jet or propeller powered, however, thrust must be passed from the user-provided subroutine. The intent of this handbook is to provide a guide to the programmer and operator of this program as well as to document the equations and assumptions used in the program. This program was adapted from the NASA takeoff and landing program documented in NASA Technical Memorandum X.622,333, (1973). This Air Force version of the program has been made to suit the needs of test and evaluation while the NASA program served as a design tool.

14. SUBJECT TERMS

simulators	simulation	computer programs
user manuals	aircraft	takeoffs
landings	continued takeoffs	refused takeoffs
TOLAND		

15. NUMBER OF PAGES
194

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

BACKGROUND

This handbook describes a general purpose computer program that will simulate the performance characteristics of aircraft takeoffs and landings. This program has been adapted from the original NASA Ames TOLAND program and documented in NASA Technical Memorandum X.62,333 (Reference 1). This original was written in FORTRAN IV in 1973 as a design tool by Jeff Bowles and Thomas Galloway. It was later modified by Dave Nesst and Wayne Olson circa 1979 to simulate refused takeoffs and rolling minimum interval takeoffs (MITOs), Reference 2. In 1980, the modified version was first used for performance simulation and flight test standardization on the F-15C flight test program and by the F-16A for landing performance tests in 1982.

Because the original NASA TOLAND was originally programmed as a design tool, it did not fully suit the needs of the flight test community. Program inputs, such as desired rate of climb, were needed as outputs; and program outputs, such as flare initiation height, were needed as inputs in a flight test oriented program. The original program ran on a large mainframe computer. As the engineers moved their flight test data processing to personal computers and workstations, a need was established to move the TOLAND program also. This new program has been completely rewritten to meet these flight test requirements. While the overall methods of numerical integration and the equations of motion have remained the same, the source code is all new.

INTRODUCTION

This document describes the inputs, the equations and computational techniques used, the subroutines and the main program developed to simulate the takeoff, landing and refused takeoff maneuvers of a given aircraft. The program can simulate takeoffs, continued takeoffs (with engine failure), refused takeoffs, and landings, and can be applied to conventional, thrust vectored and powered lift aircraft. This documentation describes the internal program subroutines, the namelist inputs, the equations and the computational techniques used.

The aircraft is treated as a point mass confined to motion in a vertical plane; the rotational dynamics have been neglected. This simplification requires an estimation of the angular rates and precludes the use of this program as a tool to evaluate minimum control speeds. These angular rates are approximated by a finite difference form or input by the user as commanded rates.

The user is required to provide at least two subroutines which pass lift coefficient (C_L), drag coefficient (C_D), net thrust (F_N), and fuel flow (W_f /dt). If random access curve files are used for aerodynamic or engine data, the initialization subroutine (INICURV) must be provided. Additional subroutines may be modified by the user for ground effect, engine transients, coefficient of friction (μ), flap retraction, gear retraction, pitch control, spoiler control, and thrust vectoring.

Detailed descriptions of the program inputs are provided in the Namelist Description section and program outputs in the Output Description section of this document. The program subroutines and functions are explained in the Subroutine Description section of this document. Detailed descriptions of the program variables contained in common blocks are provided in the Common Block Description section. The equations of motion used in the program are shown in Appendix A. Generic examples of all these subroutines are provided with the source code in Appendix B.

TAKEOFFS

The takeoff maneuver is divided into four basic segments: ground roll and rotation, liftoff and initial segment climb, constant calibrated airspeed climb, and second segment climb.

The three-point attitude ground roll is simulated at a user specified power setting, angle of attack and flap deflection. At the rotation speed (VKROTAT), the angle of attack (ALPHA) is increased linearly with time at the commanded angle of attack rate (DADTCMD) until the pitch attitude (THTROT) is reached. If the pitch attitude is reached before liftoff (before lift and the vertical component of thrust is greater than the aircraft weight), the ground roll is continued until liftoff occurs or until the takeoff ground roll time limit (ROLLMAX) is reached.

During the airborne segments of the simulation, flightpath control is obtained by controlling two dynamic variables during each integration step: acceleration along the flightpath (FPACCEL) and pitch attitude (THETAF). These variables are controlled by modulating angle of attack (ALPHA). If the flightpath acceleration drops to less than zero, or the pitch attitude exceeds the maximum pitch attitude (THTMAX), the angle of attack is reduced. The angle of attack will continue to be reduced until all these constraints are satisfied or when the rate of climb drops below zero. The simulation will terminate with an error message if the rate of climb drops below zero.

Once the initial segment climb altitude (HCLIMB) is reached, the simulation switches from tracking a constant pitch attitude (THTCLM) to tracking a constant calibrated airspeed (VCLMOUT). Once the climbout altitude (HCLMOUT) is reached, the simulation switches from tracking a constant calibrated airspeed to tracking the constant pitch segment climb gradient attitude (THTFLY). These pitch changes are performed at commanded angle of attack rate (DADTCMD) until the pitch angle is captured within the pitch angle tolerance (THTTOL). However, the simulation will terminate if the maximum altitude (HMAX), the end airspeed (VKEND), the time limit (TIMEMAX), or the ground distance limit (DISTMAX), is reached first. Table 1 shows the tracking variable (which variable the simulation is trying to match) and altitude ranges for each airborne segment of the takeoff maneuver.

Time, ground distance, and airspeed are provided when the aircraft reaches rotation speed and lifts off to the nearest integration step size time (DTIME) second, and reaches the clearance height (HCLEAR).

Table 1
TAKEOFF TRACKING VARIABLES

Climb Segment	Tracking Variable	FORTRAN Name	Altitude Range
Initial	Pitch Angle	THTCLM	0 - HCLIMB
Constant Airspeed	Calibrated Airspeed	VCLMOUT	HCLIMB- HCLMOUT
Constant Pitch Angle	Pitch Angle	THTFLY	HCLMOUT- HMAX

The "normal mode" for a takeoff is:

HCLIMB = HMAX = 50 Feet
HCLMOUT > 50 feet

in which case the program never executes the first or second segments of the climbout.

LANDINGS

The landing maneuver is divided into four basic segments: approach, flare, transition, and landing ground roll. A steady-state approach is initiated using one of two methods. In one method, the landing approach is simulated at the angle of attack and net thrust required for zero acceleration along and normal to the flightpath, e.g. at an input constant calibrated airspeed. Using the other method, the landing approach is simulated at the calibrated airspeed and net thrust required for zero acceleration along and normal to the flightpath, e.g. at an input angle of attack. These values for angle of attack or calibrated airspeed and net thrust are calculated in subroutine STEDYST using inputs of angle of attack (ALPHA) or approach speed (VKAPP), and flightpath angle (GAMMAPP). The approach starts at the obstacle clearance altitude, (HCLEAR) and ends at the flare altitude (HFLARE).

At the flare altitude, a pull-up is simulated at a normal load factor calculated based on a circular arc trajectory and the thrust is changed from the steady-state condition to idle. This trajectory is defined by the approach flightpath angle (GAMMAPP), the sink rate at touchdown (SINKTD), and the approach speed, (VKAPP). Because airspeed can change during the flare, touchdown airspeed is not an input variable. Table 2 shows the tracking variables and altitude ranges for the various segments of the landing maneuver.

Table 2
LANDING TRACKING VARIABLES

Segment	Tracking Variable	Altitude Range
Approach	Flightpath Angle and either Angle of Attack or Calibrated Airspeed	HCLEAR - HFLARE
Flare	Sink Rate at Touchdown	HFLARE - 0
Transition	Pitch Angle	0
Ground Roll	Not Applicable	0

After touchdown, aerobraking can be accomplished by pitching the aircraft to the angle of attack for aerobraking (AOAABRK), and then achieving a three-point attitude after reaching the minimum speed for aerobraking (VKABRK). The angle of attack for a three-point attitude is defined by the constant (AOA3PT). Pitch changes are simulated at a constant rate (DADTCMD) the commanded alpha rate.

During the landing ground roll, the simulation coasts the aircraft to the Wheel Brake Airspeed (VKBRAKE). In addition, individual time delays can be set for brake application (TIMEBRK), flap retraction initiation (TIMEFLP), thrust reverser deployment initiation (TIMEREV), speedbrake deployment initiation (TIMESBK) and spoiler deployment initiation (TIMESPL).

REFUSED TAKEOFFS

The refused takeoff maneuver is identical to the takeoff maneuver with the following exceptions. An engine failure is simulated at the engine failure speed (VKFAIL). The engine failure speed must be less than the liftoff speed. After a time delay from engine failure (TIMEIDL), the simulation initiates the reduction of thrust on the remaining engines to idle. After a time delay from engine failure (TIMEBRK), the simulation applies brakes by changing the current coefficient of friction (XMU) from the rolling coefficient of friction (ROLLMU) to the braking coefficient of friction (BRAKMU). If the rotation speed has been reached, the simulation will either return the angle of attack to zero or change the angle of attack to the angle of attack for aerobraking (AOAABRK). The angle of attack for aerobraking will be used for calibrated airspeeds above the aerobrake limit airspeed (VKABRK). Changes in angle of attack are made at the commanded angle of attack rate (DADTCMD). The minimum control groundspeed (VKMCG), can be used as a trigger speed to adjust engine thrust if desired. The (VKMCG) variable might be used to trigger the variable XENG to be changed to XENG + 1, on a four-engine aircraft performing a three-engine takeoff. Two engines would be used from brake release to the ground minimum control speed and then changed to three engines above that speed. This option was used on the B-1B TOLAND program to aid in developing a three-engine takeoff capability and is an uncommon use of the program.

Wheel braking can also be initiated at the wheel brake airspeed (VKBRAKE). In addition, individual time delays can be set for flap retraction initiation (TIMEFLP), thrust reverser deployment initiation (TIMEREV), speedbrake deployment initiation (TIMESBK) and spoiler deployment initiation (TIMESPL).

Engine failures can be simulated with one of three methods. The simplest method is an abrupt engine failure where the thrust loss is instantaneous. This is accomplished by setting the engine failure mode (FAILMOD) to 'SEIZE' and the delta time for engine failure (DTFAIL). These are the default settings for the program. The second method is a linear loss of thrust over a time interval. This is accomplished by setting the engine failure mode (FAILMOD) to 'SPOOL' and the delta time for engine failure (DTFAIL) to desired value in seconds.

The third method is a loss of thrust using a user-provided subroutine SPOOLDNF. This is accomplished by setting the engine failure mode (FAILMOD) to 'SPOOL' and the delta time for engine failure (DTFAIL) to 0.0. Subroutine SPOOLDNF might consist of a curve or set of curves or an exponential equation or set of exponential equations. A set of curves or a set of equations might be used to distinguish between different failure states and different initial thrust levels. SPOOLDNF is used for the spool down of the failed engine whereas an additional and identical subroutine, SPOOLDNR, is used for the spool down of the remaining engines.

Different curves might be needed to simulate a failure to idle thrust and a failure to off. The failure state (FAILST) would be used to distinguish between the different curves. Different curves might be needed if an engine failure started from military thrust or maximum continuous thrust instead of maximum thrust or takeoff rated thrust. The power code (PWRCODE), engine group (ENGGRP) or engine multiplicative factor (XENG) variables could be used to accomplish these simulation variations.

Engine failures occurring in different positions can be simulated using the engine failure group (FAILGRP) variable. This would allow different engine curve lookups for an outboard engine inoperative or an inboard engine inoperative. This variable is user-definable.

CONTINUED TAKEOFFS

The continued takeoff maneuver is identical to the takeoff maneuver with the following exceptions. An engine failure occurs at the engine failure speed (VKFAIL). The engine failure speed must be less than the liftoff speed. The minimum control groundspeed (VKMCG) can be used as a trigger speed to adjust engine thrust if desired. The VKMCG variable might be used to trigger the variable XENG to be changed to XENG + 1 on a four-engine aircraft performing a three-engine takeoff. Two engines would be used from brake release to the ground minimum control speed and then changed to three engines above that speed. This option was used on the B-1B TOLAND program to aid in developing a three-engine takeoff capability and is an uncommon use of the program.

Engine failures can be simulated with one of three methods. The simplest method is an abrupt engine failure where the thrust loss is instantaneous. This is accomplished by setting the engine failure mode (FAILMOD) to 'SEIZE' and the delta time for engine failure (DTFAIL). These are the default settings for the program. The second method is a linear loss of thrust over a time interval. This is accomplished by setting the engine failure mode (FAILMOD) to 'SPOOL' and the delta time for engine failure (DTFAIL) to desired value in seconds.

The third method is a loss of thrust using a user-provided subroutine SPOOLDNF. This is accomplished by setting the engine failure mode (FAILMOD) to 'SPOOL' and the delta time for engine failure (DTFAIL) to 0.0. Subroutine SPOOLDNF might consist of a curve or set of curves or an exponential equation or set of exponential equations. A set of curves or a set of equations might be used to distinguish between different failure states and different initial thrust levels. SPOOLDNF is used for the spool down of the failed engine whereas an additional and identical subroutine, SPOOLDNR, is used for the spool down of the remaining engines. SPOOLDNR would not be executed for a continued takeoff.

Different curves might be needed to simulate a failure to idle thrust and a failure to off. The failure state (FAILST) would be used to distinguish between the different curves. Different curves might be needed if an engine failure started from military thrust or maximum continuous thrust instead of maximum thrust or takeoff rated thrust. The power code (PWRCODE), engine group (ENGGRP) or engine multiplicative factor (XENG) variables could be used to accomplish these simulation variations.

Engine failures occurring in different positions can be simulated using the engine failure group (FAILGRP) variable. This would allow different engine curve lookups for an outboard engine inoperative or an inboard engine inoperative. This variable is user-definable.


VISTA
048
MATV

NAMELIST DESCRIPTIONS

This section describes the namelists and variables for input to the TOLAND program. Namelist DATA is used for the primary aircraft program inputs whereas namelists TKO and LND are used for the primary inputs to takeoff and landing maneuvers respectively. Namelist ROL is used for the primary inputs to the decelerating ground roll segment of a refused takeoff or a landing maneuver. Namelists DATA2, TKO2, LND2 and ROL2 are used for secondary or auxiliary inputs. These namelists are used to separate the primary inputs from the secondary inputs for ease of use and clarity. These secondary inputs would contain variables which are used infrequently or not at all by most users. Namelist TKOARY is used for inputs to flap and thrust vectoring schedules during a takeoff maneuver. Table 3 indicates the required namelist inputs for each maneuver.

Table 3
NAMELISTS

Namelist	Takeoff	Landing	Refused Takeoff	Continued Takeoff
DATA	X	X	X	X
DATA2	X	X	X	X
TKO	X		X	X
TKO2	X		X	X
TKOARY	X		X	X
LND		X		
LND2		X		
ROL		X	X	
ROL2		X	X	

Namelist DATA

CGPCT is the longitudinal position of the center of gravity as a percentage distance of the mean aerodynamic chord aft from the leading edge of the wing. The default for CGPCT is user defined in subroutine FORCEX. This input is required only if the aerodynamics are provided as a function of longitudinal position of the center of gravity.

DADTCMD is the commanded angle of attack rate. It is the rate at which the aircraft rotates during the takeoff maneuver. DADTCMD affects the rate of convergence to target pitch attitudes and target airspeeds. The default for DADTCMD is 2.5 degrees per second.

DTEMPF is the delta temperature from standard day in degrees Fahrenheit. The default for DTEMPF is 0.0 degree Fahrenheit.

FLAP is the current flap deflection in degrees during program execution. The default for FLAP is -1.0 degree. This default value allows FLAP to be set from FLPPCT. FLAP is a required input if FLPPCT is not used.

GAMMARW is the slope of the runway in degrees. The default for GAMMARW is 0.0 degree.

GWT0 is the initial gross weight in pounds. GWT0 is a required variable; there is no default.

HCLEAR is the obstacle clearance height in feet. The default for HCLEAR is 50.0 feet.

HRUNWAY is the pressure altitude of the runway in feet. The default for HRUNWAY is 0.0 feet.

VKWIND is the headwind component of wind speed in knots. Tailwinds are input using negative headwind components. The default is 0.0 knot (no wind).

Namelist DATA2

AOA3PT is a angle of attack for a 3-point attitude in degrees. The default for AOA3PT is 0.0. This variable is not required.

CONFIG is a number representing the configuration of the aircraft. This floating point variable is a user definable input. This variable is not required.

DCDX is the user drag coefficient increment. The user may alter the drag coefficient (CD) by adding or subtracting a delta drag coefficient increment. The default for DCDX is 0.0000.

DCLX is the user lift coefficient increment. The user may alter the lift coefficient (CL) by adding or subtracting a delta lift coefficient increment. The default for DCLX is 0.0000.

DTFAIL is the delta time for the failed engine to lose thrust by the thrust level defined by XENGFLD. Thrust and fuel flow decrease linearly over the time interval DTFAIL. The default for DTFAIL is 0.0 second.

DTIME is the Runge-Kutta numerical integration step size in seconds. The default for DTIME is 0.10 second. This variable controls the accuracy of the integrated variables.

ENGGRP is the operating engine group. It is a user definable three-character string variable whose contents define which engines are operating during a maneuver. Examples of ENGGRP are 'AEO', 'OEI', 'IEI', and 'AEI' which correspond to all engines operating, outboard engine inoperative, inboard engine inoperative and all engines idling respectively. The default for ENGGRP is 'AEO' for all takeoffs and 'AEI' for all landings.

EPR is engine pressure ratio. This parameter can optionally be used to input engine thrust levels to the program or can be used as a variable which contains the current value of Engine Pressure Ratio. The default for EPR is 0.0.

FAILGRP is the fail engine group. It is a user definable three-character string variable whose contents define the state of the engines after an engine failure or failures. The default for FAILGRP is 'OEI', an outboard engine inoperative.

FAILMOD is the engine failure mode. It is a five-character string variable whose contents define the mode of the failed engine. Usually, only two states are used, 'SEIZE' and 'SPOOL'. The default for FAILMOD is 'SEIZE'.

FAILST is the engine failure state. It is a four-character string variable whose contents define the state of the failed engine. Usually, only two states are used, 'IDLE' and 'OFF ', however, for aircraft with afterburners the state 'MIL ' is available. These states are quantified by the variables XIDLE, 0.0, and XMIL, respectively. The values of XIDLE and XMIL are initialized in subroutine FORCEX. The default for FAILST is 'IDLE'.

FGPCT is the gross thrust percentage increment. The user may increase the net thrust by a given percentage using this input variable. The default for FGPCT is 0.0 percent. For example, with FGPCT set to -3.5, the variable THRUST is multiplied by 0.965.

FLPPCT is the current flap percentage setting during program execution. This variable provides the user an additional method of entering flap deflection into the program. The default is user defined in subroutine FORCEX. For example, FLPPCT could be set to 50.0. The user would provide a method of initializing the FLAP variable to the corresponding value for 50 percent flaps. This variable is not required.

FLT is the flight number. It is a floating point variable available to the user for management and tracking of data. The default for FLT is 0.0. This variable is not required.

FLTNDX is the flight index. This floating point variable is a user definable input. It is not passed through any program supplied common blocks. This variable is not required.

IDBUG is the UFTAS debug code. It can provide debug information concerning the UFTAS random access curve file subroutines. The default for IDBUG is 0.

JDEBUG is the TOLAND debug code. This code is user definable; it can be used to output debug data from the user provided subroutines. Two values for JDEBUG are already used within the program. JDEBUG = 9999 for full debug output and JDEBUG = 6666 for special landing and ground roll debug output for refused takeoffs. The default for JDEBUG is 0.

LOADING is a number representing the external store loading of the aircraft. This integer variable is a user definable input. This variable is not required.

LUMSG is the logical unit for message output. It is an integer variable which determines which file certain program messages are written to. The messages that can be re-routed come from the debug output as well as UFTAS random access curve file subroutines. The default for LUMSG is LUOUT.

PWRCODE is the aircraft power code. It can describe the thrust settings to which all the aircraft engines are set. The default for PWRCODE is user defined in subroutine FORCEX.

RC is the engine thrust rating code. It can describe the thrust setting to which an engine deck or a curve file based on an engine deck is set during a simulation. The default for RC is user defined in subroutine FORCEX.

REVFLAG is the reverse thrust flag. This flag is set to .TRUE. when reverse thrust is to be enabled for a simulation. The default for REVFLAG is .FALSE.

ROLLMU is the rolling coefficient of friction. The default for ROLLMU is 0.025 and is based on MIL-C-005011B. (Reference 3)

SPDBRK0 is the initial speed brake deflection in degrees. The default for SPDBRK0 is 0.0.

THRCRV is the thrust curve type. It is a three-character input variable that is used to describe the type of Thrust Curve to be utilized. This input variable allows the program to distinguish between Thrust Curves containing different independent (Input) variables. Examples of THRCRV are 'RC', (rating code), PLA, (power lever angle) and 'EPR', (engine pressure ratio). The default for THRCRV is user defined in subroutine FORCEX.

VKMCG is the minimum control airspeed on the ground. It is an input variable used to trigger a spoolup of an operating but asymmetric engine. For example, it would be used in the case of a three-engine takeoff of a four-engine aircraft whose nose wheel steering was not sufficient to maintain directional control with full

or partial asymmetric thrust. VKMCG is not used to fail an engine; use VKFAIL. The default for VKMCG is 0.0 knot calibrated airspeed.

XENG is the engine multiplicative factor. Total engine parameters (e.g. net thrust, fuel flow) are defined by their corresponding per engine values multiplied by XENG. This variable is not an integer variable and is not the number of engines. It is a floating point variable that is used with per engine values of thrust and fuel flow. This allows the program to represent values of partial engine thrust to transition from one thrust setting to another. The default for XENG is user defined in subroutine FORCEX.

XENGFLD is the failed engines multiplicative factor. This variable is not an integer variable and is not the number of failed engines. It is a floating point variable that is used with per engine values of thrust and fuel flow. This allows the program to represent engine values of a partial failed engine (e.g. afterburner). If XENGFLD is not equal to 1.0, FAILST must be set to 'NULL'. FAILST overrides XENGFLD. The default for XENGFLD is 1.0.

Namelist TKO

HGEAR is landing gear retraction height in feet above the takeoff point. The default for HGEAR is 205 feet.

HMAX is the termination height limit in feet above the takeoff point for the takeoff simulation. The simulation will end if this limit is reached. The default for HMAX is 1000 feet.

THTCLM is the pitch attitude the simulation tracks after liftoff but before reaching HCLIMB feet altitude above the takeoff point. The default for THTCLM is 10 degrees.

THTROT is the pitch attitude the simulation tracks after rotation but before liftoff. In general, the aircraft will liftoff before reaching THTROT and will continue to rotate to THTCLM. The default for THTROT is 10 degrees.

VKFAIL is the engine failure airspeed in knots. The default for VKFAIL is 0.0 knot calibrated airspeed which is defined as no engine failure.

VKROTAT is the rotation airspeed in knots. The default for VKROTAT is 0.0 knot calibrated airspeed. The simulation will rotate the aircraft without regard to actual aircraft horizontal tail authority; TOLAND does not define moments about the aircraft center of gravity.

Namelist TKO2

DISTMAX is ground distance limit for the takeoff simulation. The simulation will end if this limit is reached. The default for DISTMAX is 60,760 feet, or ten nautical miles.

HCLIMB is the height in feet above the takeoff point at which the takeoff simulation switches from tracking a constant pitch attitude (THTCLM) to tracking a constant calibrated airspeed (VCLMOUT). The default for HCLIMB is 100 feet.

HCLMOUT is the height in feet above the takeoff point at which the takeoff simulation switches from tracking a constant calibrated airspeed (VCLMOUT) to tracking a constant pitch attitude (THTFLY). The default for HCLMOUT is 200 feet.

ROLLMAX is the takeoff ground roll time limit. The simulation will end if this limit is reached. The default for ROLLMAX is 120 seconds.

THTFLY is the pitch attitude the simulation tracks after reaching HCLMOUT feet altitude above the takeoff point. The default for THTFLY is 8 degrees.

THTTOL is the pitch attitude tolerance the simulation tracks to. The simulation tracks to the target pitch angle within plus or minus THTTOL degrees; after which no pitch adjustment is made. The default for THTTOL is 0.10 degree.

TIMEMAX is the time limit for the takeoff simulation. The simulation will end if this limit is reached. The default for TIMEMAX is 300 seconds.

TKOTYPE is the takeoff type. It is a seven character input variable that is used to describe the type of takeoff as either 'STATIC' or 'ROLLING'. This input variable allows the program to distinguish between the thrust settings of either a static or rolling takeoff independently of the VKSTART input variable. The default for TKOTYPE is 'STATIC'.

VCLMOUT is the calibrated airspeed (in knots) the simulation tracks after reaching HCLIMB feet altitude above the takeoff point but before reaching HCLMOUT feet altitude above the takeoff point. The default for VCLMOUT is the calibrated airspeed reached at HCLIMB feet altitude above the takeoff point.

VKEND is the simulation end airspeed in knots. The default for VKEND is 250 knots calibrated airspeed.

VKFLAP is the calibrated airspeed at which the flaps are retracted. The default for VKFLAP is the flap limit airspeed, VKFLPMX. The flap limit airspeed, VKFLPMX, is a hard coded value user defined in subroutine 'FXXAERO'.

VKSTART is the simulation start groundspeed in knots. This input variable allows the user to simulate takeoffs where there is a given groundspeed at the time the aircraft is aligned on the centerline of the runway. For example, rolling minimum interval takeoffs (MITOs) frequently roll onto the centerline of the runway without stopping. The default for VKSTART is 0.0 knot.

Namelist TKOARY

This namelist contains parameters that are not required.

FLPARY is the flap deflection array. This array contains five elements. This array allows the user to enter in a flap deflection schedule as a function of calibrated airspeed. At each airspeed contained in the flap deflection airspeed array (VFLPARY), the takeoff simulation changes the flap deflection, FLAP, to the corresponding flap deflection contained in the flap deflection array (FLPARY) at a rate of DFLAPDT degrees per second. The flap deflection rate (DFLAPDT) is provided by subroutine 'FXXAERO' which is the user provided subroutine called by subroutine FORCEX. If the last element of FLPARY is not 0.0, the simulation will retract the flaps to 0.0 degree at the calibrated airspeed contained in the last element of VFLPARY or at VKFLPMX which ever is less. The default for the elements of FLPARY a.e 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the flap deflection until the flap retraction speed (VKFLAP) or the maximum flap deflection airspeed (VKFLPMX) is reached.

VFLPARY is the flap deflection airspeed array. This array contains five elements. This array allows the user to enter in a flap deflection schedule as a function of calibrated airspeed. At each airspeed contained in the flap deflection airspeed array (VFLPARY) the takeoff simulation changes the flap deflection (FLAP) to the corresponding flap deflection contained in the flap deflection array (FLPARY) at a rate of DFLAPDT degrees per second. The flap deflection rate (DFLAPDT) is provided by subroutine 'FXXAERO' which is the user provided subroutine called by subroutine FORCEX. If the last element of FLPARY is not 0.0, the simulation will retract the flaps to 0.0 degree at the calibrated airspeed contained in the last element of VFLPARY or at VKFLPMX which ever is less. The default for the elements of VFLPARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the flap deflection until the flap retraction speed (VKFLAP) or the maximum flap deflection airspeed (VKFLPMX) is reached.

HVCTARY is the vectored thrust altitude array. This array contains five elements. At each altitude contained in the vectored thrust altitude array, HVCTARY, the takeoff simulation changes the vectored thrust angle (VTANGLE), to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is the user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of HVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle.

VVCTARY is the vectored thrust airspeed array. This array contains five elements. At each calibrated airspeed contained in the vectored thrust airspeed array (VVCTARY) the takeoff simulation changes the vectored thrust angle (VTANGLE) to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is the user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of VVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle.

XNUARY is the vectored thrust angle array. This array contains five elements. At each altitude contained in the vectored thrust altitude array (HVCTARY) or at each calibrated airspeed contained in the vectored thrust airspeed array (VVCTARY) the takeoff simulation changes the vectored thrust angle (VTANGLE) to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is the user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of VVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle.

Namelist LND

ALPHA is the angle of attack in degrees during approach. The program will determine the calibrated airspeed (VKAPP) for the approach if ALPHA is input. ALPHA or VKAPP are required variables; there are no defaults.

GAMMAPP is the flightpath angle in degrees (negative for a descent) during approach. The default for GAMMAPP is -3.0 degrees (descending).

HFLARE is the height in feet above the touchdown point of the runway at which the simulation initiates a flare. The default for HFLARE is 0.0 feet (no flare).

SINKTD is the sink rate at touchdown. The default for SINKTD is 10.0 feet per second.

VKAPP is the calibrated airspeed in knots during the approach. The program will determine the angle of attack (ALPHA) for the approach if VKAPP is input. VKAPP or ALPHA are required variables; there are no defaults.

Namelist LND2

DTDTMX is the maximum pitch rate during the flare simulation in degrees per second. The default for DTDTMX is user defined in subroutine FORCEX.

SPLFLAG is the spoiler flag. The flag is set to .TRUE. to initialize the simulation with the spoilers deployed. The default for SPLFLAG is .FALSE. (spoilers retracted).

WRITITR is the write flare iterations flag. This flag is set to .TRUE. when the user desires output on each flare iteration executed from subroutine FLARENZ. The default for WRITITR is .FALSE.

Namelist ROL

BRAKMU is the braking coefficient of friction. This input parameter would be used if the braking coefficient selector (IMU) is set to 0 (corresponding to a constant BRAKMU). The default for BRAKMU is 0.250.

IMU is the braking coefficient selector. The default selections for IMU are constant braking coefficient (BRAKMU), 0; user supplied curve file lookup, 1; generic dry runway braking coefficient, 2; and generic wet runway braking coefficient, 3. The default for IMU is 0, which corresponds to a constant braking coefficient of friction. For IMU=1, BRAKMU might be looked up as a function of weight on wheels, groundspeed and runway condition reading (RCR). The default for BRAKMU is 0.250.

RCR is the runway condition reading. It can be used as an input to determine the variable BRAKMU. The default for RCR is user defined in subroutine GENMU.

Namelist ROL2

AOAABRK is the angle of attack for aerobraking during the landing ground roll if the calibrated airspeed is greater than VKABRK knots calibrated airspeed. The default for AOAABRK is 0.0 degree (no aerobraking).

BRKFCTR is the braking factor. It is a multiplicative factor applied to BRAKMU, the braking coefficient of friction. It allows the user to adjust the coefficient of friction during braking without recompiling code. This input parameter would normally be used if the braking coefficient selector (IMU) was not set to 0. The default for BRKFCTR is 1.0.

TIMEBRK is the braking time delay in seconds between the time touchdown has occurred and the time the brakes are applied for a landing maneuver. It is also the braking time delay between the time an engine has failed and the time the brakes are applied for a refused takeoff maneuver. The default for TIMEBRK is 3.0 seconds.

TIMEFLP is the flap retraction time delay in seconds between the time touchdown has occurred and the time the flap retraction initiation has occurred retracting for a landing maneuver. It is also the flap retraction time delay between the time an engine has failed and the time the flap retraction initiation has occurred for a refused takeoff. The default for TIMEFLP is 999.0 seconds; essentially the flaps are not retracted.

TIMEIDL is the idle thrust spool down time delay in seconds between the time an engine has failed and the time a spool down to idle has been initiated on the remaining engine(s) for a refused takeoff. This parameter has no effect during a landing because the engines are in idle during the flare. The default for TIMEIDL is 3.0 seconds.

TIMEREV is the reverse thrust time delay in seconds between the time touchdown has occurred and the time the thrust reverser deployment initiation has occurred for a landing maneuver. It is also the reverse thrust time delay between the time the remaining engine(s) has spooled down to idle and the time the thrust reverser deployment initiation has occurred for a refused takeoff maneuver. The default for TIMEREV is 0.0 second.

TIMESBK is the speedbrake deployment time delay in seconds between the time touchdown has occurred and the time the speedbrake deployment initiation has occurred for a landing maneuver. It is also the speedbrake time delay between the an engine has failed and the time the speedbrake deployment initiation has occurred for a refused takeoff maneuver. The default for TIMESBK is 0.0 second.

TIMESPL is the spoiler deployment time delay in seconds between the time touchdown has occurred and the time the spoiler deployment initiation has occurred for a landing maneuver. It is also the spoiler time delay between the an engine has failed and the time the spoiler deployment initiation has occurred for a refused takeoff maneuver. The default for TIMESPL is 0.0 second.

VKABRK is the calibrated airspeed during the landing ground roll that aerobraking is stopped and the angle of attack (which is the same as pitch angle during the ground roll) is lowered from AOAABRK to 0.0 degree. The default for VKABRK is 0.0 knot calibrated airspeed (no aerobraking).

VKBRAKE is the wheel braking airspeed in knots. Wheel braking is initiated after the calibrated airspeed is less than VKBRAKE. The default for VKBRAKE is 999.0 knots calibrated airspeed.

OUTPUT DESCRIPTIONS

This section describes the output provided by the TOLAND program.

Takeoff and Continued Takeoff Output

For each set of namelist inputs provided, the program echoes many of the input variables back to the output file. Then the core of the program output is written. This core consists of the following variables listed in a table from left to right: elapsed time (TIME), ground distance (GDIST), gross weight (GWT), altitude above ground level (HAGL), knots calibrated airspeed (VKCAS), knots true airspeed (VKTAS), knots true groundspeed (VKTGS), acceleration along flightpath or ground track (FPACCEL or ACCEL), lift coefficient (CL), drag coefficient (CD), pitch attitude (THETAF), angle of attack (ALPHA), flightpath angle (GAMMA), time rate of change of angle of attack (DADT), rate of climb (ROC), normal load factor (XLF), net thrust (THRUST), and the engine multiplicative factor (XENG). The core data is repeated once every 10 DTIME seconds. However at rotation, liftoff, and the obstacle clearance height the data is also listed.

After the core output is written, summary information about the takeoff is output. This summary information consists of flight number (FLT), initial gross weight (GWT0), user drag coefficient increment, (DCDX), net thrust percentage increment (FGPCT), ground minimum control airspeed (VKMCG), original flap setting (FLAP0), delta temperature from standard day (DTEMPF), user lift coefficient increment (DCLX), the failure state (FAILST), and the operating engine group (ENGGRP).

Refused Takeoff Output

The output for refused takeoffs is identical to that for takeoffs and continued takeoffs with the following exception: the coefficient of friction (XMU), is substituted for the engine multiplicative factor (XENG), during the decelerating portion of the ground roll.

Landing Output

The core output for landings is identical to that for takeoffs and continued takeoffs with the following exceptions: the time rate of change in pitch attitude (DTDT), is substituted for the time rate of change in angle of attack (DADT), during the approach and flare portion of the landing; and the coefficient of friction (XMU), is substituted for the engine multiplicative factor (XENG), during the decelerating portion of the landing ground roll.

After the core output is written, summary information about the landing is output. This summary information consists of the distance over an HCLEAR foot obstacle, air distance, ground roll distance, pre-flare distance, flare distance, and average deceleration. Figure 1 defines these distances for a landing with the obstacle clearance height (HCLEAR), greater than the flare height (HFLARE); a low flare landing.

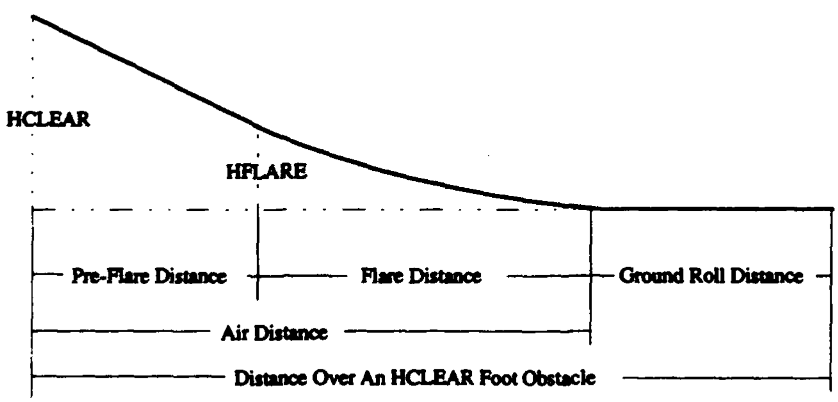


Figure 1 Low Flare Landing Distance Definitions

Figure 2 defines these distances for a landing with the obstacle clearance height (HCLEAR), less than the flare height (HFLARE); a high flare landing. In addition, flight number (FLT), user drag coefficient increment (DCDX), delta temperature from standard day (DTEMPF), braking multiplicative factor (BRKFCTR), user lift coefficient increment (DCLX), and original flap setting (FLAP0) are also output.

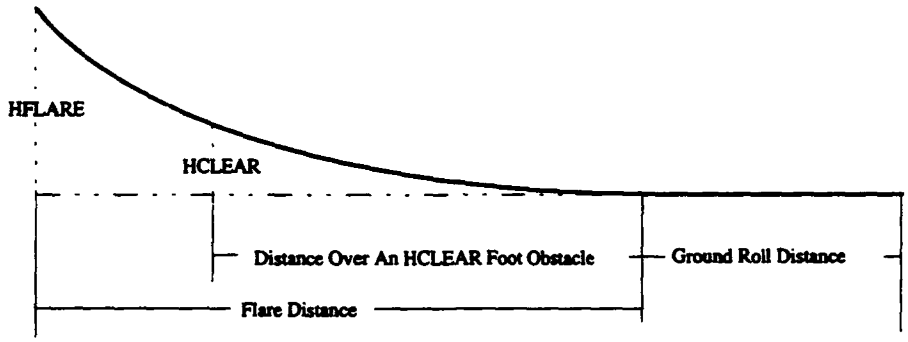


Figure 2 High Flare Landing Distance Definitions

PROGRAM STRUCTURE

This section describes the structural layout of the TOLAND program.

All execution begins and ends with the main program. Namelist inputs are read and parameter initialization are performed first. Curve file initialization is then performed (if applicable). Program execution then branches to one of two control subroutines, TAKOFF or LANDNG, as shown in Figure 3. For takeoffs, refused takeoffs or continued takeoffs, execution branches to subroutine TAKOFF. For landings, execution branches to subroutine LANDNG.

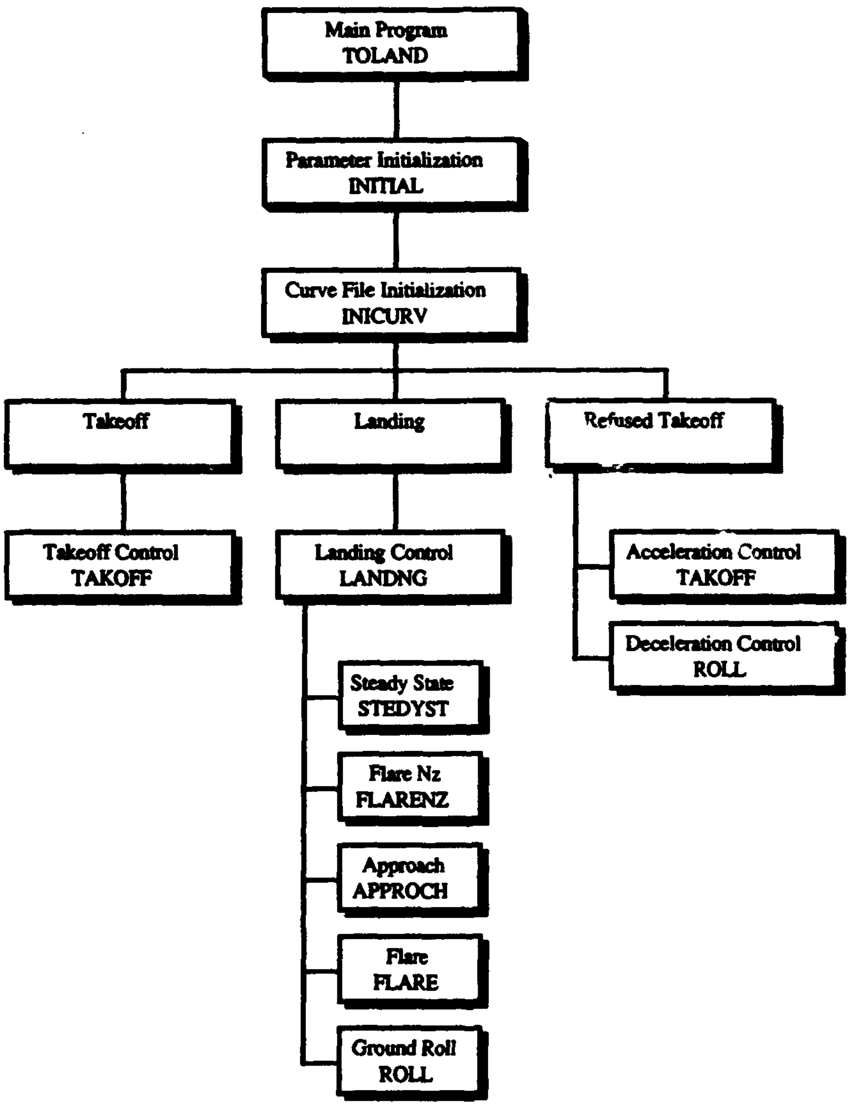


Figure 3 Takeoff and Landing Program Structural Layout

Control is passed to subroutine TAKOFF for takeoffs and to subroutine LANDNG for landings. Control is first passed to subroutine TAKOFF for refused takeoffs, until engine failure when control is passed to

subroutine ROLL. After the maneuver is complete, program execution returns to the main program. The main program then either reads another set of namelist inputs or terminates execution.

Within subroutine LANDNG, program execution passes to subroutine STEDYST to determine the steady-state conditions for the landing approach. After that is accomplished, program execution is passed to subroutine FLARENZ, where the constant normal load factor for the flare (if any) is determined through multiple calls to subroutine FLARE. Program execution then passes to subroutine APPROCH (if the obstacle clearance height (HCLEAR) is greater than the flare initiation height (HFLARE)) or subroutine FLARE (if the obstacle clearance height (HCLEAR) is less than or equal to the flare initiation height (HFLARE)). After touchdown, program execution passes to subroutine ROLL until the end of the ground roll. Program execution then passes back to subroutine LANDNG and then to the main program. The main program either reads another set of namelist inputs or terminates execution.

Runge-Kutta numerical integration is performed from only subroutines TAKOFF, FLARE, and ROLL. The integrator subroutine INTX determines the acceleration from one of equations of motion subroutines, DERIVGR, DERIVAT, or DERIVAL. The force coefficients for the equations of motion are provided by subroutine FORCEX. FORCEX calculates the force coefficients from the lift and drag coefficients provided by subroutine 'FXXAERO' and from thrust provided by subroutine 'FXXENG'. The associations of these subroutines are shown in Figure 4.

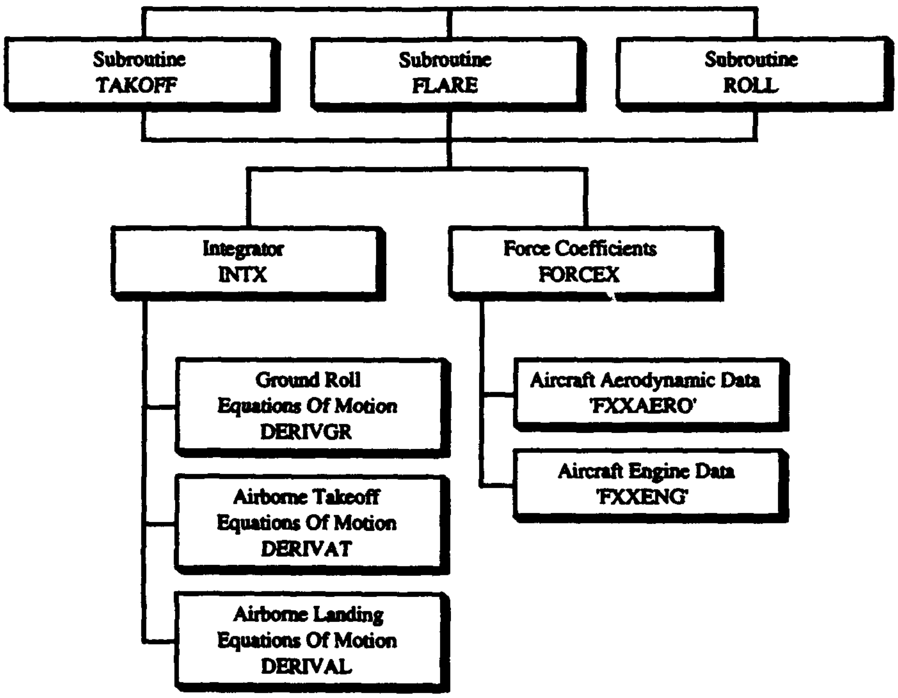


Figure 4 Takeoff and Landing Program Major Subroutine Associations

SUBROUTINE DESCRIPTIONS

This section describes the subroutines and functions of the TOLAND program.

Table 4
SUBROUTINE USAGE

Subroutine	Takeoff	Landing	Refused Takeoff	Continued Takeoff
INITIAL	P	P	P	P
FRETRAC[1]	P,C	P,C	P,C	P,C
GRETRAC[2]	P,C			P,C
PITCH	P	P	P	P
SPOIL[3]		P	P	
TVECTOR	O	O	O	O
TAKOFF	P		P	P
LANDNG		P		
STEDYST		P		
FLARENZ[4]		P		
APPROCH[5]		P,C		
FLARE		P		
ROLL		P	P	
INTX	P	P	P	P
DERIVGR	P	P	P	P
DERIVAT	P			P
DERIVAL		P		
ERROR[6]	P,C			P,C
HALT	P	P	P	P
ATMOSPH	P	P	P	P
ITRLND		P		
DGDT		P		
DVDT		P		
DVTDH		P		
DADH		P		
DDELTDH		P		
INTERP	P	P	P	P
ZEROX		P		
INICURV	U	U	U	U
FORCEX	U	U	U	U
'FXXAERO'	U	U	U	U
GEFFECT	U	U	U	U
'FXXENG'	U	U	U	U
SPOOLUP[7]	U,C	U,C	U,C	U,C
SPOOLDNF[7]		U,C	U,C	U,C
SPOOLDNR[7]		U,C	U,C	U,C
GENMU		U,C	U,C	

C Conditional execution of subroutine
O Optional sub-routine-- not required
P Program provided subroutine
U User provided subroutine

NOTES:
1. FRETRAC is called only when the flap setting is changed.
2. GRETRAC is called only when gear is retracted.
3. SPOIL is called only when the spoiler setting is changed.
4. FLARENZ is called when HFLARE is greater than 0.
5. APPROCH is called when HCLEAR is greater than HFLARE.
6. ERROR is called when the equations of motion subroutine is unable to find a solution.
7. SPOOLUP, SPOOLDNF, or SPOOLDNR is called when a throttle transient is performed.

Table 4 shows which subroutines are executed for which maneuver. Conditional subroutines are executed only if the certain conditions are met. See notes (1-7). The optional subroutine TVECTOR is not required if thrust vectoring is not used. Program provided subroutines are intended to be used without modifications. User provided subroutines have program supplied templates for easy modification.

User Provided Subroutines

Some of the following subroutines must be modified for a particular aircraft. For aircraft simulations using random access curve files, subroutine INICURV must be modified to contain the architecture of the user's curves and initializes the appropriate arrays in common blocks VALUES and TABLES. Subroutine FORCEX requires initializing certain aircraft unique constants within a DATA statement and using the proper calling arguments in calls to subroutines 'FXXAERO' and 'FXXENG'

Subroutine 'FXXAERO' must return lift coefficient (C_L), and drag coefficient (C_D) to subroutine FORCEX. At least the lift coefficient must be a function of angle of attack (α); this program is angle of attack driven. Subroutine GEFFECT can be used for ground effect increments to the lift and drag coefficients. A sample subroutine is provided that includes Dr. E K Parks increment equations.

Subroutine 'FXXENG' must return gross thrust (F_G), propulsive drag (F_E), and fuel flow (W_f /dt) to subroutine FORCEX. Samples of subroutines SPOOLDNF, SPOOLDNR, and SPOOLUP are provided as guides to the user for simulating throttle transients.

Subroutine GENMU must return the braking coefficient (BRAKMU). A sample is provided as a guide to the user showing a variety of methods which produce braking coefficient.

Subroutine INICURV
This subroutine initializes the names, tables and value names of the random access curve files. This subroutine opens the appropriate curves as needed. This subroutine has no calling arguments and contains common blocks: FLAGS, CHARV, CINDEX, RACURV, VALUES, and TABLES.

Subroutine FORCEX
This subroutine calculates the total force coefficients along and normal to the flightpath. Equations of motion defining the force coefficients are shown in Appendix A. The force coefficients are calculated with the lift and drag coefficients supplied by subroutine 'FXXAERO' and with thrust supplied by subroutines 'FXXENG', SPOOLDNF, SPOOLDNR and SPOOLUP.

Subroutine FORCEX should require only the following modifications:

1) Initialization of the DATA statement providing values for AIT, AOA3PT, AR, B, CGPCT, CLALPH, CONFIG, DTDTMX, FLAP, FLPPCT, HZ, LOADING, NENG, PWRCODE, RC, SWING, THRCRV, THTMAX, XIDLE, and XLFMAX. DTDTMX and XLFMAX is only required for the flare portion of landings. AR, B, CLALPH, and HZ are variables used only in determining ground effect increments to the aerodynamic models. CONFIG and LOADING are optional user-definable variables. PWRCODE, RC, and THRCRV are optional user-definable variables for distinguishing power settings and switching between different engine curves.

2) Changing the calling statement for subroutines 'FXXAERO' and 'FXXENG' to the user provided names with the appropriate calling arguments.

This subroutine has the calling arguments: ALPHA, CD, and CL; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRBORN, AIRSPED, FLAGS, and CHARV.

Subroutine 'FXXAERO'
This subroutine calculates or performs a table lookup to determine the lift and drag coefficients for the aircraft. This subroutine and its name are provided by the user. For example, this subroutine might be called F22AERO for the F-22. The calling arguments are user defined but might be ALPHA, AMACH, CL, CD, FLAP, SPOILER, VKCAS and XENGOUT. The subroutine might contain common blocks: CTRL,

AIRCRFT, AIRSPED, RUNWAY, FPINTEG, ATMOS, FLAGS, CHARV, RACURV, VALUES and TABLES.

Subroutine GEFFECT
This subroutine supplies the ground effect increments to the aerodynamic coefficients. The generic subroutine provides predicted ground effect equations which can be modified by the user for a particular aircraft. The calling arguments are user defined but might be CL, HAGL, DCLGE and DCDGE. The subroutine might contain common blocks: AIRCRFT, AERO, CONST, and CHARV.

Subroutine 'FXXENG'
This subroutine calculates or performs a table lookup to determine the net thrust and fuel flow for the aircraft. This subroutine and its name are provided by the user. For example, this subroutine might be called F22ENG for the F-22. The calling arguments are user defined but might be ALPHA, AMACH, PWRCODE, QS, VKCAS, XENG, FE, FG, THRUST, WFUEL, and EPR. The subroutine might contain the common blocks: CTRL, VECTOR, RUNWAY, FPINTEG, ATMOS, CONST, FLAGS, CHARV, RACURV, VALUES and TABLES.

Subroutine SPOOLUP
This subroutine determines the ratio of thrust to takeoff rated thrust during a rolling takeoff using subroutine TABINT and data array SPOOLA. This is accomplished by passing the engine multiplicative factor, XENG, back to the calling subroutine. This subroutine is just an example and does not apply to all aircraft. This subroutine has the calling arguments: ENGNDX, TIME, XENGTRN, XENG0, SPOOL and XENG; and contains no common blocks.

Subroutine SPOOLDNF
This subroutine determines the engine multiplicative factor for a failed engine during a throttle chop to idle or a fuel cut using subroutine TABINT and data array SPOOLA. This subroutine is just an example and does not apply to all aircraft. This subroutine has the calling arguments: TIME, XENGEND, XENGTRN, SPOOL, XENG, and LUMSG and contains the common block: CHARV.

Subroutine SPOOLDNR
This subroutine determines the engine multiplicative factor for the remaining engines during a throttle chop to idle or a fuel cut using subroutine TABINT and data array SPOOLA. This subroutine is just an example and does not apply to all aircraft. This subroutine has the calling arguments: TIME, XENGEND, XENGTRN, SPOOL, XENG, and LUMSG and contains the common block: CHARV.

Subroutine GENMU
This subroutine provides the braking coefficient of friction. This subroutine has the calling arguments: VKTGS, WTML, XMAIN, XNOSE and YCG; and contains the common blocks: CTRL, RUNWAY, CONST, RACURV, VALUES, and TABLES. The user must provide values for XMAIN, XNOSE, and YCG. These are all positive values which correspond to distances from the aircraft's center of gravity. XMAIN is the horizontal distance from the main gear to the center of gravity. XNOSE is the horizontal distance from the nose gear to the center of gravity. YCG is the vertical distance from the ground to the center of gravity.

Program Subroutines

Subroutine INITIAL
This subroutine initializes groups of variables. This subroutine has the calling arguments: GROUP, ALPHA, FLAP0, GWT0, and SPDBRK0; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, VECTOR, AIRBORN, AIRSPED, RUNWAY, INTEG, FPINTEG, ATMOS, CONST, FLAGS, CHARV, FLAPDAT, GEARDAT, VECTDAT, and RACURV.

Subroutine FRETRAC
This subroutine controls the flap setting during flap retraction. This subroutine has the calling arguments: FLAP, VKFLAP, and DFLAPDT; and contains common blocks: CTRL, FLAGS, and FLAPDAT.

Subroutine GRETRAC
This subroutine returns the value of the incremental drag of the landing gear, DCDLGR. The flag LGRFLAG is available to add the incremental drag in subroutine 'FXXAERO'. This subroutine has the calling arguments: DCDLGR, DTGEAR and HAGL; and contains common blocks: CTRL, FLAGS, and GEARDAT.

Subroutine PITCH
This subroutine modulates angle of attack to match an input pitch attitude or climb speed if limit conditions are met. The rate at which angle of attack is increased is based on normal load factor (XLF) and the commanded alpha rate (DADTCMD), a user input. This subroutine has the calling arguments: MANUVR, ALPHA, DADTCMD, DTIME, DTDTGEX, and LUOUT; and contains common blocks: AIRCRFT, AIRBORN, FPINTEG, CONST and FLAGS.

Subroutine SPDBRAK
This subroutine controls speed brake deployment and retraction. This subroutine has the calling arguments: ACTION, SPDBRK, SBKEND, and DSBKDT and contains common blocks: CTRL and FLAGS. The possible values for action are 'RESET', 'DEPLOY', and 'RETRACT'.

Subroutine SPOIL
This subroutine controls spoiler deployment and retraction. This subroutine has the calling arguments: ACTION, SPOILER, SPLREND, and DSPLRDT and contains common blocks: CTRL and FLAGS. The possible values for action are 'RESET', 'DEPLOY', and 'RETRACT'.

Subroutine TVECTOR
This subroutine controls the vectored thrust angle (VTANGLE). This subroutine has the calling arguments: VTANGLE, HVECT, VVECT, and DVECTDT; and contains common blocks: CTRL, FLAGS, and VECTDAT.

Subroutine TAKOFF
This subroutine controls the execution of the takeoff maneuver from brake release through climb out. Subroutine TAKOFF calls INTX to perform numerical integration of the resultants of the equations of motion and calls FORCEX to obtain the force coefficients for the equations of motion. This subroutine has the calling argument: ALPHA; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRBORN, AIRSPED, RUNWAY, INTEG, FPINTEG, ATMOS, CONST, FLAGS, CHARV, FLAPDAT, GEARDAT, and VECTDAT.

Subroutine LANDNG
This subroutine controls the execution of the landing maneuver. LANDNG calls STEDYST to obtain required values of thrust and angle of attack for a steady-state approach. LANDNG calls subroutine FLARENZ to determine the normal load factor required to execute a flare at the desired sink rate at touchdown. LANDNG call subroutine APPROCH to execute an approach down to the flare height (HFLARE) if the obstacle clearance height (HCLEAR) is greater than HFLARE. LANDING calls subroutine FLARE to execute a flare and calls subroutine ROLL for the ground portion of the landing. This subroutine has the calling argument: ALPHA; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRBORN, AIRSPED, RUNWAY, INTEG, FPINTEG, ATMOS, CONST, and FLAGS.

Subroutine STEDYST
This subroutine calculates the required values of net thrust and angle of attack for zero acceleration along and normal to the flightpath, dV_c /dt and $V_c(d\gamma$ /dt) respectively when the flag, FINDV is set to .FALSE. It calculates the required values of net thrust and calibrated airspeed for zero acceleration along and normal to

the flightpath, dV_c/dt and $V_c(d\gamma/dt)$ respectively when the flag, FINDV is set to .TRUE. Finding one of these two sets of values is accomplished by driving two functions of two independent variables to zero simultaneously

$$f(F_N,\alpha) = dV_c/dt = 0$$
$$g(F_N,\alpha) = V_c(d\gamma/dt) = 0$$

for all non-zero positive values of V_c, this reduces to

$$g(F_N,\alpha) = (d\gamma/dt) = 0$$

where:

F_N is net thrust
α is angle of attack
V_c is calibrated airspeed
dV_c/dt is the time rate of change of velocity along flightpath
$d\gamma/dt$ is the time rate of change of flightpath angle

Subroutine STEDYST starts generating the locus of $f(F_N,\alpha) = 0$ and $g(F_N,\alpha) = 0$, and then searches for values of net thrust and angle of attack, or net thrust and calibrated airspeed, which satisfy the intersection of the two loci within a tolerance bound. The output values of net thrust and angle of attack, or net thrust and calibrated airspeed, are the required values for a steady-state approach.

Figure 5 shows a typical plot of the dV_c/dt and $d\gamma/dt$ functions, f and g, respectively. The values α_{app} and $F_{N_{app}}$ are the desired values of angle of attack and net thrust. The upper bound on thrust is the gross weight (GWT) divided by the engine multiplicative factor (XENG). The search for α_1 and α_2 is made on the interval $-\alpha_{max} \le \alpha_i \le \alpha_{max}$, i=1,2. The error is then defined to be the difference between α_1 and α_2 divided by α_1. Subroutine STEDYST then varies the value of net thrust until the error is driven to less than the subroutine tolerance, or $\alpha_1 - \alpha_2 \le \varepsilon_t$. This tolerance value for α, ε_t, is 0.005 degrees until the number of iterations within the subroutine exceeds 25; then ε_t is raised to 0.010 degrees. The initial value of angle of attack is estimated based on the lift coefficient where lift equals gross weight. The initial value of net thrust is estimated to be 1.06 times the sum of the drag at the initial angle of attack and the excess thrust (net thrust). Net thrust is varied each iteration, until bounded, by the following equation: $F_{N_{i+1}} = F_{N_i}/1.05$. Making this initial 1.1 times the net thrust prevents the program searching in areas where no solution to $dV_t/dt = 0$ exists. For finding calibrated airspeed the process is identical. Calibrated airspeed is substituted for angle of attack. However, the tolerance value for Vc, ε_t, is 0.001 knot until the number of iterations within the subroutine exceeds 25; then ε_t is raised to 0.005 knot.

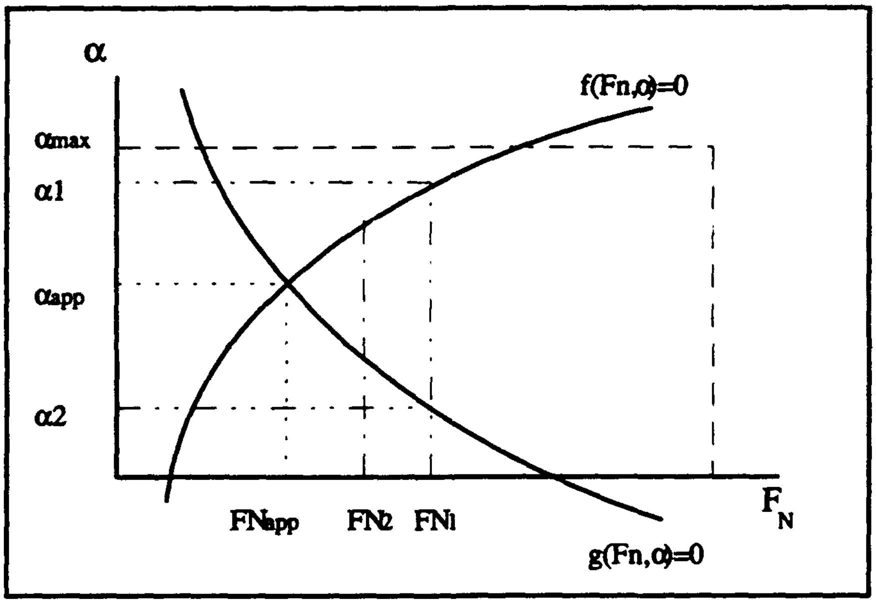


Figure 5 Steady-State Approach Solution

Subroutine STEDYST has calling arguments: ALPHA, FPVTAS, GAMMAR, and FINDV; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, ATMOS, CONST and FLAGS. The mechanization of subroutine STEDYST for determining THRUST and ALPHA is as follows. The outer loop varies THRUST, while the two inner loops vary ALPHA from ALPHMX to - ALPHMX. For a fixed value of THRUST (outer loop), functions DVDT (acceleration along the flightpath) and DGDT (acceleration normal to the flightpath), are called with the values of ALPHA (inner loops). These two functions call subroutine FORCEX. Subroutine ITRLND and function ZEROX are both zero finders called from STEDYST. Once the values of net thrust are bounded, function ZEROX locates the values within the boundaries to within a specified tolerance. The tolerance for net thrust is based on 0.1 percent of the aircraft gross weight, or 5 pounds, whichever is greater.

The loop containing the call to ITRLND is exited successfully when search status variable, JFLAG, is set internally by the subroutine to 3, or when the magnitude of the error is less than the tolerance value (TOLRNCE). Variable JCOUNT stores the number of iterations which is limited to 25.

Subroutine FLARENZ

This subroutine determines the normal load factor, n_z, of the flare maneuver. Subroutine FLARENZ calls FLARE at various normal load factors until the sink rate at touchdown (SINKTD) is matched to within EPSROC feet/minute. EPSROC is hard-coded at 0.5 feet/minute. When FLARENZ is first called, the initial guess for normal load factor is determined by assuming the flare is using the following simplified trajectory. The flare trajectory is estimated by a constant normal load factor (n_z) constant airspeed circular arc starting at the flare height (HFLARE), flightpath angle (GAMMAPP), and approach speed (VKAPP). The end of the arc is defined by the sink rate at touchdown (SINKTD), and the approach speed (VKAPP).

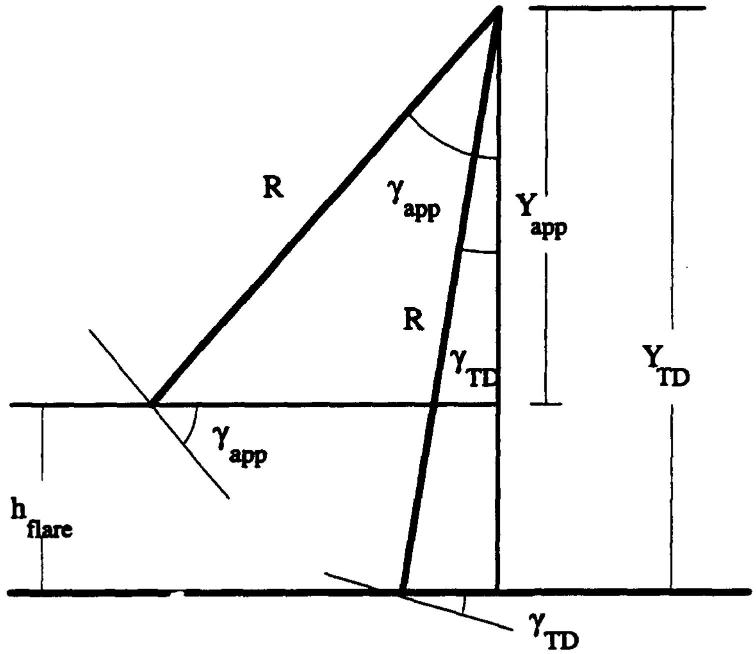


Figure 6 Initial Flare Load Factor Determination

Figure 6, shows a diagram of the flare. The flare height can be defined as a function of the flare radius (R) and the flare radius can be represented using the circular centripetal acceleration equation:

$$h_{flare} = Y_{TD} - Y_{app} = R\cos\gamma_{TD} - R\cos\gamma_{app} = R(\cos\gamma_{TD} - \cos\gamma_{app}) \quad (1)$$

$$a_n = V_{t_{app}}^2/R = g(n_z - 1.0) \quad (2)$$

Substituting equation 2 for R into equation 1,

$$h_{flare} = [V_{t_{app}}^2(\cos\gamma_{TD} - \cos\gamma_{app})/g]/(n_z - 1.0) \quad (3)$$

Solving for normal load factor as a function of the two flightpath angles and approach speed,

$$n_z = [V_{t_{app}}^2(\cos\gamma_{TD} - \cos\gamma_{app})/g\ h_{flare}] + 1.0 \quad (4)$$

This gives subroutine FLARENZ a starting point for iterating on normal load factor to match the sink rate at touchdown.

Subroutine FLARENZ uses ITRLND to determine new values of normal load factor to use in calling FLARE. FLARENZ can call subroutine FLARE up to MAXITER times. MAXITER is hardcoded at 31. This subroutine has the calling arguments: ALPHA, GAMMAPP, GAMMATD, HAGL, HCLEAR, HFLARE, HRUNWAY, SINKTD, THRUST, VKAPP, XLFLARE, and XLFMAX; and contains common blocks: CTRL, FPINTEG, CONST, and FLAGS.

Subroutine APPROCH

This subroutine controls execution of the approach above the flare height, HFLARE. This subroutine is called only if the obstacle clearance height HCLEAR is greater than HFLARE. APPROCH calculates

values for output every APPDTIM seconds. APPDTIM is hard coded to 0.5 second. This subroutine has the calling arguments: ALPHA, GAMMAPP, GDIST, HAGL, HCLEAR, HFLARE, and HRUNWAY; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRSPED, FPINTEG, ATMOS, CONST, and FLAGS.

Subroutine FLARE
This subroutine controls execution of the flare maneuver. Subroutine FLARE calls INTX to perform numerical integration of the resultants of the equations of motion and calls FORCEX to obtain the force coefficients for the equations of motion. This subroutine has the calling arguments: ALPHA, GDIST, ODIST, ROCTD, and VKTGS; and contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRBORN, AIRSPED, RUNWAY, FPINTEG, ATMOS, CONST, and FLAGS.

The variable DTDTGEX is provided to the user as a means to account for a loss in the pitch rate capability due to ground effect. DTDTGEX may be a function of wing height/wingspan; total lift coefficient (CL), or circulation lift coefficient; et cetera. DTDTGEX would equal 1.0 out of ground effect and less than 1.0 in ground effect. DTDTGEX should be calculated in subroutine 'FXXAERO' and passed through the calling statement to subroutine FORCEX. The user may ignore DTDTGEX without affecting program execution.

Subroutine ROLL
This subroutine controls the execution of the ground roll of the landing maneuver or the deceleration portion of a refused takeoff maneuver. Subroutine ROLL calls INTX to perform numerical integration of the resultants of the equations of motion and call FORCEX to obtain the force coefficient for the equations of motion. The calling arguments are, angle of attack (ALPHA) and the ground distance covered by the approach and flare or the takeoff ground roll (GDIST); ROLL contains common blocks: CTRL, AIRCRFT, AERO, ENGINE, AIRSPED, RUNWAY, INTEG, ATMOS, CONST, FLAGS and CHARV.

Subroutine INTX
This subroutine performs the numerical integration using a fourth order Runge-Kutta scheme with the Adams-Bashforth-Moulton predictor-corrector method (Reference 4). The calling arguments are NEQ, TIME, DTIME, T, DERIV and ALPHA. NEQ is the number of equation of motion to be integrated. T is either the values of common block INTEG or FPINTEG. DERIV is a subroutine that must be made external in the calling subroutine. DERIV corresponds to either DERIVGR, DERIVAT, or DERIVAL; INTX contains no common blocks.

Subroutine INTG
This subroutine calculates incremental distance, DDIST; total ground distance, GDIST; and gross weight, GWT. This subroutine has the calling arguments: DIST, DISTJ, DTIME, GAMMAR, VWIND, WFUEL, DDIST, GDIST, and GWT; and contains no common blocks.

Subroutine DERIVGR
This subroutine calculates the acceleration, ACCEL, for the ground roll of a takeoff, landing or refused takeoff. This subroutine has the calling argument: ALPHA; and contains common blocks: CTRL, AIRCRFT, AERO, AIRSPED, RUNWAY, INTEG, CONST, and FLAGS.

Subroutine DERIVAT
This subroutine calculates the time derivatives for the airborne portion of the takeoff and manages the flightpath control. This subroutine has the calling argument: ALPHA; and contains common blocks: AIRCRFT, AERO, AIRBORN, FPINTEG, CONST, and FLAGS.

Subroutine DERIVAL
This subroutine calculates the time derivatives for the airborne portion of the landing and manages the flightpath control. This subroutine has the calling argument: ALPHA; and contains common blocks: CTRL, AIRCRFT, AERO, AIRBORN, FPINTEG, CONST, and FLAGS.

Subroutine ERROR
This subroutine is called during a takeoff simulation when the flag ERRFLAG is set to .TRUE. Program execution is terminated for the present namelist inputs and will continue if additional namelist inputs are to be processed. This subroutine has the calling arguments: LUOUT and ROCFPM; and contains no common blocks.

Subroutine HALT
This subroutine terminates program execution and writes a termination message, TERMMSG. This subroutine has the calling arguments: LUIN, LUMSG, LUOUT, and TERMMSG; and contains no common blocks.

Subroutine ATMOSPH
This subroutine calculates atmospheric pressure, temperature, density, speed of sound, kinematic viscosity, as well as pressure, temperature and density ratios. The inputs are pressure altitude (PRESALT) and the temperature increment from standard day (DTEMPF). This subroutine has the calling arguments: PRESALT and ARRAY; and contains common block ATMOS.

Subroutine SPEED
This subroutine calculates dynamic pressure, Mach number, calibrated airspeed, equivalent airspeed, true airspeed, and groundspeed. This subroutine has the calling arguments: GAMMAR, VTASX, VWIND, AMACH, QS, VKCAS, VKEAS, VKTAS, VKTGS, and VTGS; and contains common blocks: AIRCRFT, ATMOS, and CONST.

Subroutine ITRLND
This subroutine is a zero-finding routine which varies the independent variable based on the sign and size of the error. The loop containing the call to ITRLND is exited successfully when JFLAG = 3 or when the magnitude of the error is less than the tolerance value, TOLRNCE. This subroutine has the calling arguments: ERROR, ERRORJ, DRIVER, FACTOR, TOLRNCE and JFLAG; and contains no common blocks. ERROR and ERRORJ are the present and previous errors, respectively. JFLAG is a variable which indicates the status of the search. JFLAG is set to 0 before ITRLND is called. After the first call to ITRLND, JFLAG is set to 1. JFLAG is set to 2 once the zero is isolated between the upper and lower boundaries, BOUNDU and BOUNDL, respectively. JFLAG is set to 3 and the zero is considered found when the absolute value of the difference between BOUNDU and BOUNDL is less than TOLRNCE, the tolerance passed to subroutine ITRLND.

During the first call to ITRLND DRIVERJ, the previous DRIVER, is set to DRIVER and then DRIVER is multiplied by FACTOR. This forms a range with boundaries, DRIVERJ and DRIVER. On subsequent calls to ITRLND, these two driver values are set to either an upper or lower boundary or a multiplication or division by FACTOR is applied to DRIVER. Once the location of the zero is determined to be on one side of the DRIVER or the other, DRIVER is set to either the upper or lower boundary as the search is narrowed.

Program Function Descriptions

Function DGDT
This function returns the acceleration normal to the flightpath. This function has the calling arguments: ALPHA, FPVTAS, GAMMAR, and GWT; and contains common blocks: AERO, AIRSPED and CONST.

Function DVDT
This function returns the acceleration along the flightpath, dV_c/dt. Function DVDT uses functions, DVTDH, DADH, and DDELTDH to account for acceleration from flying at a constant calibrated airspeed. This function has the calling arguments: ALPHA, FPVTAS, GAMMAR, and GWT; and contains common blocks: AERO, AIRSPED and CONST.

Function DVTDH
This function returns the climb speed derivative with respect to pressure altitude (dV_t/dH) for a constant calibrated airspeed, equivalent airspeed or Mach number. Using this function for a calibrated airspeed (by setting CONSTANT = 'VC') allows function DVDT to return a value dV_c/dt instead of dV_t/dt. This function has the calling arguments: VTAS and CONSTANT; and contains common blocks: CTRL, FPINTEG, ATMOS, CONST and RACURV.

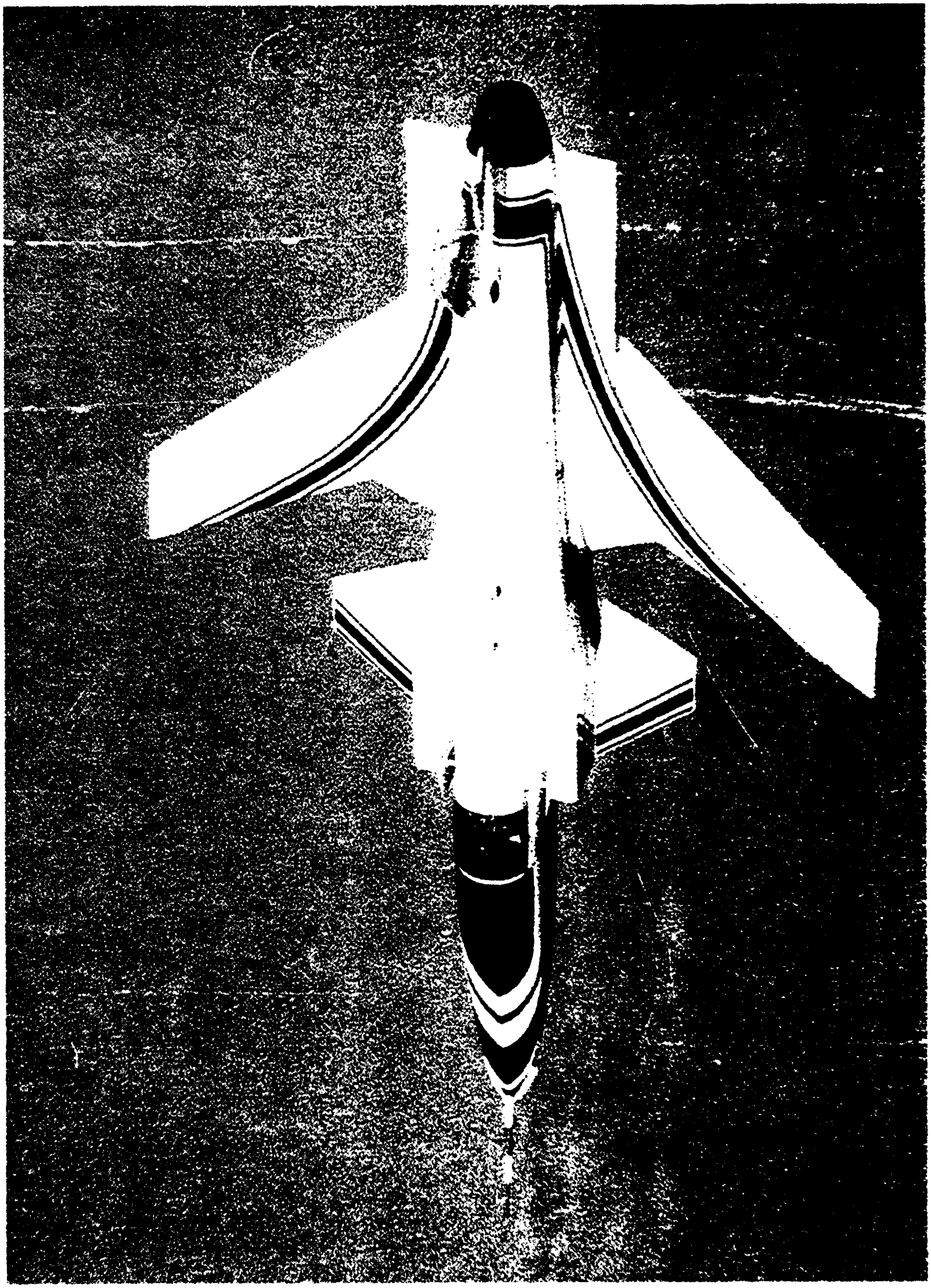
Function DADH
This function returns the speed of sound derivative with respect to pressure altitude (da/dH). This function has the calling argument: HC; and contains common block: CTRL.

Function DDELTDH
This function returns the pressure ratio derivative with respect to pressure altitude ($d\delta/dH$). This function has the calling argument: HC; and contains common block: CTRL.

Function DSIGDH
This function returns the density ratio derivative with respect to pressure altitude, $d\sigma/dH$. This function has the calling argument: HC; and contains common block: CTRL.

Function INTERP
This function returns an interpolated value between two ordinate inputs based on the additional inputs of three abscissas. To prevent an arithmetic indefinite this function resets the difference of the two maxima abscissas to one if the difference equals zero. This function has the calling arguments: YNOW, YPAST, XNOW, XPAST and XANS; and contains no common blocks.

Function ZEROX
This function finds the location (in X) of a zero of a function FUNCT(X) on an interval X = A to X = B. The function ZEROX assumes that F(A) and F(B) are of opposite sign, implying that FUNCTN(X) is equal to zero somewhere within that interval. Function ZEROX finds this zero by recursively calling function FUNCTN. ZEROX uses the logical variable, FINDV, to transfer the proper calling arguments to function FUNCTN. If FINDV is .TRUE., ZEROX passes calibrated airspeeds for A and B; otherwise, angles of attack are passed. The argument "FUNCTN" must be declared external in the calling subroutine and have four calling arguments, A or B, VAR2, VAR3, and VAR4. Functions DGDT and DVDT meet this criteria. ZEROX has the calling arguments: A, B, FUNCTN, VAR2, VAR3, VAR4, TOLRNCE, and FINDV; and contains no common blocks.



COMMON BLOCK DESCRIPTIONS

This section describes the common blocks of the TOLAND program and their variables.

Common CTRL

Common CTRL contains variables concerning program control.

DTIME is the current Runge-Kutta Numerical Integration Step Size in seconds. The default for DTIME is 0.10 seconds; DTIME is entered through namelist DATA2.

DTIMEJ is the previous Runge-Kutta numerical integration step size in seconds. This variable is used to store the original value of DTIME while the current value is lowered to allow the output to be printed at an even interval of time This variable is used at the interfaces of subroutines APPROCH, FLARE, and ROLL.

ICOUNT is the integration loop initializing variable. After switching from one integration loop to another, ICOUNT stores the integer variable which initializes the integration loop. Therefore, the program will write the output variables after executing both integration loops a total of 10 times. This could be two iterations of the first loop and eight iterations of the next loop or three iterations of the first loop and seven iterations of the next loop, et cetera. Placing ICOUNT in Common CTRL allows the program to execute different integration loops in different subroutines while still retaining writing the output variables once every 10 iterations. With DTIME, the Runge-Kutta Integration Step Size set to 0.1 seconds, output will occur once per second of simulation time.

JDEBUG is the TOLAND debug code. This code is user definable; it can be used to output debug data from the user provided subroutines. When JDEBUG is set to 6666, debug output for the steady-state iterations is printed. The JDEBUG setting of 9999 is reserved and used for full debug output. The default for JDEBUG is 0; JDEBUG is entered through namelist DATA2.

KENG is the subroutine 'FXXENG' output switch. It allows the program to switch between the types of output subroutine 'FXXENG' can provide subroutine 'FXXENG' outputs gross thrust and fuel flow from a power setting input if KENG is set equal to 1. Subroutine 'FXXENG' outputs power setting from an input of thrust if KENG is set equal to 2. This subroutine is provided by the user; KENG can be just a null or inactive variable. Currently all the calls to subroutine 'FXXENG' require outputs of thrust and fuel flow from a power setting input.

LINENUM is the current line number of the current page of output. When LINENUM exceeds NPAGE, a new page is started.

LUIN is the logical unit for input. It is an integer variable which determines which file namelist inputs are read from. The default for LUIN is 3, but can be user defined; LUIN is initialized in the main program.

LUMSG is the logical unit for message output. It is an integer variable which determines which file certain program messages are written to. The messages that can be re-routed come from the debug output as well as UFTAS random access curve file subroutines. The default for LUMSG is LUOUT; LUMSG is entered through namelist DATA2.

LUOUT is the logical unit for output. It is an integer variable which determines which file program output is written to. The default for LUOUT is 4, but can be user defined; LUOUT is initialized in the main program.

NCOUNT is the integration loop counter. After executing an integration loop 10 times, the program will write the output variables. Placing NCOUNT in common CTRL allows the program to execute different integration loops in different subroutines while still retaining writing the output variables once every

10 iterations. With DTIME, the Runge-Kutta integration step size set to 0.1 seconds, output will occur once per second of simulation time.

NEQ is the number of equations of motion to be integrated. NEQ is set to 2 for the takeoff ground roll and landing roll and is set to 4 for the airborne portions of the takeoffs and landings.

NPAGE is the number of lines of output per page. When LINENUM exceeds NPAGE, a new page is started. The default for NPAGE is 64 lines per page; NPAGE is hard coded in the main program.

TIME is the elapsed time in seconds.

TIMEROL is the elapsed time of the ground roll. This elapsed time corresponds to the landing ground roll or the portion of the ground roll of a refused takeoff after engine failure.

Common AIRCRFT

Common AIRCRFT contains many of the aircraft specific constants and variables.

AOA3PT is the angle of attack for the aircraft in a three-point attitude. This aircraft constant is used to initialize angle of attack at the beginning of a takeoff and to set the zero angle of attack flag (AOA0FLG) to true for a landing. AOA3PT is initialized in subroutine FORCEX.

AR is the aspect ratio of the wing. This aircraft constant is used to calculate ground effect variables. AR is initialized in subroutine FORCEX.

B is the wingspan in feet. This aircraft constant is used to calculate ground effect variables. B is initialized in subroutine FORCEX.

CGPCT is the longitudinal position of the center of gravity as a percentage distance of the mean aerodynamic chord aft from the leading edge of the wing. The default for CGPCT is user defined in subroutine FORCEX; CGPCT is entered through namelist DATA.

CLALPH is the lift curve slope out of ground effect. This aircraft constant is used to calculate ground effect variables. Because this constant is used only in the user provided subroutine, GEFFECT, it can have units of either 1/degrees or 1/radians. CLALPH is initialized in subroutine FORCEX.

CONFIG is a number representing the configuration of the aircraft. This floating point variable is a user definable input. CONFIG is initialized in subroutine FORCEX. This variable is not required; CONFIG is entered through namelist DATA2.

DTDTMX is the maximum pitch rate during the landing simulation in degrees per second. The default for DTDTMX is user defined in subroutine FORCEX; DTDTMX is entered through namelist LND2.

FLT is the flight number. It is a floating point variable available to the user for management and tracking of data. The default for FLT is 0.0; FLT is entered through namelist DATA2.

GWT is the gross weight during program execution in pounds. GWT is a required variable; there is no default. GWT is initialized as GWT0. GWT0 is entered through namelist DATA.

HZ is the distance from the bottom of the wing to the bottom of the tires in feet. This aircraft constant is used to calculate ground effect variables. HZ is initialized in subroutine FORCEX.

LOADING is a number representing the external store loading of the aircraft. This integer variable is a user definable input. LOADING is initialized in subroutine FORCEX. This variable is not required; LOADING is entered through namelist DATA2.

SWING is the wing surface area in square feet. SWING is initialized in subroutine FORCEX.

THTMAX is the maximum pitch attitude limit in degrees. THTMAX is initialized in subroutine FORCEX.

WNGLOD is the wing loading in pounds per square foot. It is calculated in the main program and output at the beginning of a takeoff simulation in subroutine TAKOFF.

XLFMAX is the simulation maximum normal load factor in g's. XLFMAX is initialized in subroutine FORCEX. XLFMAX is only used for the flare portion of a landing.

Common AERO

Common AERO contains the major aerodynamic parameters.

CX is the force coefficient along the flightpath or ground roll. This parameter is further defined in Appendix A, Equations of Motion.

CY is the force coefficient normal to the flightpath or ground roll. T[illegible] parameter is further defined in Appendix A, Equations of Motion.

DADTCMD is the commanded angle of attack rate. It is the rate at which the aircraft rotates during the takeoff maneuver. DADTCMD affects the rate of convergence to target pitch attitudes and target airspeeds. The default for DADTCMD is 2.5 degrees per second; DADTCMD is entered through namelist DATA.

DCDX is the user drag coefficient increment. The user may alter the drag coefficient (CD) by adding a delta drag coefficient increment. The default for DCDX is 0.0000; DCDX is entered through namelist DATA2.

DCLX is the user lift coefficient increment. The user may alter the lift coefficient (CL) by adding a delta lift coefficient increment. The default for DCLX is 0.0000; DCLX is entered through namelist DATA2.

DTDTGEX is the loss in pitch rate capability due to ground effect. The user may lower the pitch rate during the flare in ground effect by using this multiplicative factor applied to pitch rate. The default for DTDTGEX is 1.0; DTDTGEX can be calculated in subroutine 'FXXAERO' and passed through the calling statement to subroutine FORCEX.

FLAP is the current flap deflection in degrees during program execution. The default for FLAP is -1.0 degree. This default value allows FLAP to be set from FLPPCT. FLAP is a required input if FLPPCT is not used. FLAP is entered through namelist DATA.

FLPPCT is the current flap percentage setting during program execution. This optional variable provides the user an additional method of entering flap deflection into the program. The default is user defined in subroutine FORCEX. FLPPCT is entered through namelist DATA2.

QS is the product of current dynamic pressure (Q) and wing surface area (SWING). The units for QS are in pounds.

SPDBRK is the current speed brake deflection in degrees.

SPOILER is the current spoiler deflection in degrees.

VKCAS is the current calibrated airspeed in knots.

VKTAS is the current true airspeed in knots.

Common ENGINE

Common ENGINE contains the major engine parameters.

AIT is the thrust incidence angle in degrees. This constant is initialized in subroutine FORCEX.

AMACH is the current Mach number.

DTFAIL is the delta time for the failed engine to lose thrust by the thrust level defined by XENGFLD. Thrust and fuel flow decrease linearly over the time interval DTFAIL. The default for DTFAIL is 0.0 second; DTFAIL is entered through namelist DATA2.

EPR is engine pressure ratio. This parameter can optionally be used to input engine thrust levels to the program or can be used as a variable which contains the current value of Engine Pressure Ratio. The default for EPR is 0.0; EPR is entered through namelist DATA2.

FE is the current propulsive drag of all the engines in pounds.

FG is the current gross thrust of all the engines in pounds.

FGPCT is the gross thrust percentage increment. The user may increase the gross thrust by a given percentage using this input variable. The default for FGPCT is 0.0 percent; FGPCT is entered through namelist DATA2.

NENG is the total number of engines the aircraft has. This constant is initialized in subroutine FORCEX.

PWRCODE is the aircraft power code. It can describe the thrust settings to which all the aircraft engines are set. The default for PWRCODE is user defined in subroutine FORCEX; PWRCODE is entered through namelist DATA2.

REVNDX is the reverse engine spool-up index. It can describe the spool-up curve in subroutine SPOOLUP which corresponds to thrust reversers. This constant is user defined in subroutine 'FXXENG' and must be passed through as a calling argument to subroutine FORCEX; REVNDX is not required if thrust reversers are not to be simulated.

THRUST is the current total net thrust of all the engines in pounds.

VTANGLE is the current vectored thrust angle in degrees.

WFUEL is the current total fuel flow of all the engines in pounds per hour.

XENG is the current engine multiplicative factor. Total engine parameters (e.g. net thrust, fuel flow) are defined by their corresponding per engine values multiplied by XENG. The default for XENG is user defined in subroutine FORCEX; XENG is entered through namelist DATA2.

XENGFLD is the failed engines multiplicative factor. This variable is not an integer variable and is not the number of failed engines. It is a floating point variable that is used with per engine values of thrust and fuel flow. This allows the program to represent engine values of a partial failed engine (e.g. afterburner). If XENGFLD is not equal to 1.0, FAILST must be set to 'NULL'. FAILST overrides XENGFLD. The default for XENGFLD is 1.0; XENGFLD is entered through namelist DATA2.

XENGOUT is the engine multiplicative factor after an engine failure of XENGFLD engines. It is defined as the XENG - XENGFLD.

XIDLE is the idle engine multiplicative factor for a single engine. The default for XIDLE is 0.06; XIDLE is user defined in subroutine FORCEX.

XMIL is the military thrust engine multiplicative factor for a single engine. The default for XMIL is 0.50; XMIL is user defined in subroutine FORCEX.

ZFN is a user definable variable.

Common AIRBORN

Common AIRBORN contains airborne control and status parameters.

ALPHAJ is the previous angle of attack calculated from the last integration in degrees.

ALPHMX is the maximum angle of attack limit in degrees. ALPHMX is set to THTMAX in subroutine LANDNG.

DTDT is the current pitch rate in degrees per second.

GAMMAPP is the flightpath angle in degrees during approach. The default for GAMMAPP is -3.0 degrees (descending); GAMMAPP is entered through namelist LND.

ROCFPM is the current rate of climb in feet per minute.

THETAF is the current pitch attitude in degrees.

XLF is the current normal load factor in g's.

XLFJ is the previous normal load factor calculated from the last numerical integration in g's.

Common AIRSPED

Common AIRSPED contains the airspeed constants used for program control.

VKABRK is the calibrated airspeed during the landing ground roll that aerobraking is stopped and the angle of attack is lowered from AOAABRK to 0.0 degree. The default for VKABRK is 0.0 knot calibrated airspeed (no aerobraking); VKABRK is entered through namelist ROL2.

VKAPP is the calibrated airspeed in knots during the approach. VKAPP is entered through namelist LND.

VKBRAKE is the wheel braking airspeed in knots. Wheel braking is initiated after the calibrated airspeed is less than VKBRAKE. The default for VKBRAKE is 999 knots calibrated airspeed; VKBRAKE is entered through namelist ROL2.

VKEND is the simulation end airspeed in knots. The default for VKEND is 250 knots calibrated airspeed; VKEND is entered through namelist TKO2.

VKFAIL is the engine failure airspeed in knots. The default for VKFAIL is 0.0 knot calibrated airspeed which is defined as no engine failure; VKFAIL is entered through namelist TKO.

VKFLAP is the calibrated airspeed at which the flaps are retracted. The default for VKFLAP is the flap limit airspeed (VKFLPMX); VKFLAP is entered through namelist TKO2. The flap limit airspeed (VKFLPMX) is user defined in subroutine 'FXXAERO'.

VKFLPMX is the maximum flap calibrated airspeed in knots. This constant is used as the default for VKFLAP; it is user defined in subroutine 'FXXAERO'.

VKMCG is the minimum control airspeed on the ground. It is an input variable used to trigger a spool-up of an operating but asymmetric engine. It would be used in the case of (for example) a three-engine takeoff of a four-engine aircraft whose gross weight was not sufficient for nose wheel steering to account for the full or partial asymmetric thrust. VKMCG is not used to fail an engine; use VKFAIL. The default for VKMCG is 0.0 knot calibrated airspeed; VKMCG is entered through namelist DATA2.

VKROTAT is the rotation airspeed in knots. The default for VKROTAT is 0.0 knot calibrated airspeed. The simulation will rotate the aircraft without adequate tail authority; TOLAND runs with only engine and aerodynamic models which do not define moments about the aircraft center of gravity; VKROTAT is entered through namelist TKO.

VKSTART is the simulation start groundsoeed in knots. This input variable allows the user to simulate takeoffs where there is a given groundspe at the time the aircraft is aligned on the centerline of the runway. The default for VKSTART is 0.0 knot; VKSTART is entered through namelist TKO2.

VKWIND is the headwind component of wind speed in knots. Tailwinds are input using negative wind speeds. The default for VKWIND is 0.0 knot (no wind); VKWIND is entered through namelist DATA.

VWIND is the headwind component of wind speed in feet per second. Tailwinds use negative wind speeds.

Common RUNWAY

Common RUNWAY contains the major parameters related to the runway.

ABARG is the average deceleration in g's

AOAABRK is the angle of attack for aerobraking during the landing ground roll if the calibrated airspeed is greater than VKABRK knots calibrated airspeed. The default for AOAABRK is 0.0 degree (no aerobraking); AOAABRK is entered through namelist ROL2.

BRAKMU is the braking coefficient of friction. This input parameter would be used if IMU, the Braking Coefficient Selector, is set to 0 (corresponding to a constant BRAKMU). The default for BRAKMU is 0.250.; BRAKMU is entered through namelist ROL.

BRKFCTR is the braking factor. It is a multiplicative factor applied to BRAKMU, the Braking Coefficient of Friction. It allows the user to adjust the coefficient of friction during braking without recompiling code. This input parameter would normally be used if IMU, the Braking Coefficient Selector, was not set to 0. The default for BRKFCTR is 1.0; BRKFACTR is entered through namelist ROL2.

GAMMARW is the slope of the runway in degrees. The default for GAMMARW is 0.0 degree; GAMMARW is entered through namelist DATA2.

GRW is the slope of the runway in radians.

HAGL is the current altitude of the aircraft above the liftoff point on the runway for takeoff simulations or the current altitude of the aircraft above the runway for landing simulations.

HCLEAR is the obstacle clearance height in feet. The default for HCLEAR is 50.0 feet; HCLEAR is entered through namelist DATA.

HFLARE is the height in feet above the touchdown point of the runway at which the simulation initiates a flare. The default for HFLARE is 0.0 feet (no flare); HFLARE is entered through namelist LND.

HGEAR is landing gear retraction height in feet above the takeoff point. The default for HGEAR is 205 feet; HGEAR is entered through namelist TKO.

HRUNWAY is the pressure altitude of the runway in feet. The default for HRUNWAY is 0.0 feet; HRUNWAY is entered through namelist DATA.

IMU is the braking coefficient selector. The default selections for IMU are constant braking coefficient, BRAKMU, 0; user supplied curve file lookup, 1; generic dry runway braking coefficient, 2; and generic wet runway braking coefficient, 3. The default for IMU is 0, which corresponds to a constant braking coefficient of friction, BRAKMU; IMU is entered through namelist ROL. The default for BRAKMU is 0.250; BRAKMU is entered through namelist ROL.

RCR is the runway condition reading. It can be used as an input to determine the variable BRAKMU. The default for RCR is user defined in subroutine GENMU; RCR is entered through namelist ROL.

ROLLMU is the rolling coefficient of friction. The default for ROLLMU is 0.025; ROLLMU is entered through namelist DATA2.

TIMEBRK is the braking time delay in seconds between the time touchdown has occurred and the time the brakes are applied for a landing maneuver. It is also the braking time delay between the time an engine

has failed and the time the brakes are applied for a refused takeoff maneuver. The default for TIMEBRK is 3.0 seconds; TIMEBRK is entered through namelist ROL2.

TIMEFLD is the elapsed time from engine failure. TIMEFLD is initialized in subroutine TAKOFF when the engine failure speed (VKFAIL) is attained.

TIMEFLP is the flap retraction time delay in seconds between the time touchdown has occurred and the time the flap retraction initiation has occurred retracting for a landing maneuver. It is also the flap retraction time delay between the time an engine has failed and the time the flap retraction initiation has occurred for a refused takeoff. The default for TIMEFLP is 999.0 seconds; essentially the flaps are not retracted. TIMEFLP is entered through namelist ROL2.

TIMESBK is the speedbrake deployment time delay in seconds between thc time touchdown has occurred and the time the speedbrake deployment initiation has occurred for a landing maneuver. It is also the speedbrake time delay between the an engine has failed and the time the speedbrake deployment initiation has occurred for a refused takeoff maneuver. The default for TIMESBK is 0.0 second; TIMESBK is entered through namelist ROL2.

TIMESPL is the spoiler deployment time delay in seconds between the time touchdown has occurred and the time the spoiler deployment initiation has occurred for a landing maneuver. It is also the spoiler time delay between the an engine has failed and the time the spoiler deployment initiation has occurred for a refused takeoff maneuver. The default for TIMESPL is 0.0 second; TIMESPL is entered through namelist ROL2.

XMU is the current coefficient of friction. It is set equal to either ROLLMU or BRAKMU. ROLLMU is a constant entered through namelist DATA. BRAKMU is a variable set in subroutine GENMU or entered as a constant though namelist ROL.

Common INTEG

Common INTEG contains the ground roll integration variables.

VTAS is the true airspeed in feet per second; it is the integral of ACCEL with respect to time.

DIST is the air distance (ground distance uncorrected for wind) in feet; It is the integral of VTAS with respect to time.

ACCEL is the ground roll acceleration in feet per second squared. ACCEL is provided by the equation of motion executed in subroutine DERIVGR.

VTASJ is the previous true airspeed in feet per second. It is used to continually initialize the Runge-Kutta array for the ground roll iterations.

RKGRND is an array containing previous values of the ground roll integration variables. These values are used in the fourth order Runge-Kutta numerical integration calculations to determine future values.

Common FPINTEG

Common FPINTEG contains the flightpath integration variables.

FPVTAS is the flightpath true airspeed in feet per second; it is the integral of FPACCEL with respect to time.

GAMMAR is the flightpath angle in radians; it is the integral of DGDTR with respect to time.

FPDIST is the flightpath air distance in feet; it is the integral of FPVTAS with respect to time.

PRESALT is the pressure altitude in feet; it is the sum of HRUNWAY and the integral of ROC with respect to time.

FPACCEL is the flightpath acceleration in feet per second squared. FPACCEL is provided from an equation of motion executed in either subroutine DERIVAT or subroutine DERIVAL.

DGDTR is the rate of change in flightpath angle with respect to time in radians per second. DGDTR is provided by an equation of motion executed in either subroutine DERIVAT or subroutine DERIVAL.

VHAS is the horizontal airspeed in feet per second. VHAS is calculated from FPVTAS and GAMMAR in either subroutine DERIVAT or subroutine DERIVAL.

ROC is the rate of climb in feet per second. ROC is calculated from FPVTAS and GAMMAR in either subroutine DERIVAT or subroutine DERIVAL.

RKAIR is an array containing previous values of the flightpath integration variables. These values are used in the fourth order Runge-Kutta numerical integration calculations to determine future values.

Common ATMOS

Common ATMOS contains the atmospheric parameters.

TEMPR is the current temperature in degrees Rankine.

PRESS is the current ambient pressure in pounds per square foot.

RHO is the current density is slugs per cubic feet.

AFPS is the current speed of sound in feet per second.

VISCOSK is the current kinematic viscosity in feet squared per second.

DELTA is the current pressure ratio.

SIGMA is the current density ratio.

THETA is the current temperature ratio.

DTEMPF is the delta temperature from standard day in degrees Fahrenheit. The default for DTEMPF is 0.0 degree Fahrenheit; it is entered through namelist DATA.

Common CONST

Common CONST contains the major program constants.

ASL is the speed of sound at sea level in knots. This constant is equal to 661.48 and is initialized in the main program.

ASLSQR5 is the speed of sound at sea level multiplied by the square root of 5.0, = 1479.114245. This constant is used in the calculation for calibrated airspeed.

FPSKTS is the conversion factor from knots to feet per second. This constant is equal to 1.687806 and is initialized in the main program.

G is the reference acceleration at sea level, 45 degrees latitude, due to gravity. This constant is equal to 32.174 and is initialized in the main program.

RX is the conversion factor from radians to degrees. This constant is equal to 57.2957759130824 and is initialized in the main program

TSLF is the standard day temperature at sea level in degrees Fahrenheit. This constant is equal to 59.0 degrees and is initialized in the main program.

TWOOVR7 is the ratio of 2.0 over 7.0. This constant is used in the calculation for calibrated airspeed; it is equal to 0.2857142857142857 and is initialized in the main program.

ZERO is the constant 0.0; it is initialized in the main program.

Common FLAGS

Common FLAGS contains the major variables of type LOGICAL.

AOA0FLG is the zero angle of attack flag. This flag is .TRUE. when the angle of attack (ALPHA) is equal to 0.0.

BRKFLAG is the brake flag. This flag is set to .TRUE. when the conditions for brake application are met. The conditions are: current calibrated airspeed (VKCAS) less than brake application speed (VKBRAKE) and elapsed time from engine failure or touchdown (TIMEROL) is greater than the braking time delay (TIMEBRK).

CLRHGT is the clearance height flag. This flag is set to .TRUE. after the current altitude (HAGL) exceeds the clearance height (HCLEAR).

ERRFLAG is the error flag. This flag is set to .TRUE. by any of the equation of motion subroutines, (DERIVGR, DERIVAT, or DERIVAL) when certain flightpath constraints are not satisfied and the condition is not recoverable. The program will terminate processing with the current set of namelist inputs.

FAILFLG is the engine failure flag. This flag is set to .TRUE. when a refused takeoff or a continued takeoff simulation is to be performed. At the engine failure speed (VKFAIL) the engine failure speed flag (VFFLAG) is set to .TRUE. and the program switches the operating engine group (ENGGRP) to the fail engine group (FAILGRP). If the engine failure state (FAILST) is set to 'OFF' the current engine multiplicative factor (XENG) is set to the engine multiplicative factor after engine failure (XENGOUT). If the time for a failed engine to lose thrust, DTFAIL, is greater than zero, the current engine multiplicative factor is linearly reduced to the engine multiplicative factor after engine failure, XENGOUT, over DTFAIL seconds.

FLAPFLG is the flap flag. This flag is .TRUE. when the flap deflection, FLAP, is greater than zero.

FPCTFLG is the flap percentage flag. This flap is set to .TRUE. when the flap deflection is initialized at -1. With the flap percentage flag set to .TRUE., the flap percentage variable (FLPPCT) is used to determine flap deflection (FLAP).

GEFLAG is the ground effect flag. This flag is set to .TRUE. when the aircraft is in ground effect. The conditions which determine when this flag should be .TRUE. are to be user defined.

LGRFLAG is the landing gear flag. This flag is .TRUE. when the landing gear is not fully retracted.

LIFTOFF is the liftoff flag. This flag is set to .TRUE. after liftoff is achieved.

OVERFLG is the flare height over obstacle clearance height flag. This flag is set to .TRUE. if the flare height (HFLARE) is greater than the obstacle clearance height (HCLEAR).

REVFLAG is the reverse thrust flag. This flag is set to .TRUE. when reverse thrust is to be enabled for a simulation.

REVRSE is the thrust reverse active flag. This flag is set to .TRUE. when reverse thrust is simulated.

ROTATE is the rotation flag. This flag is set to .TRUE. after rotation is initiated.

RTOFLAG is the refused takeoff flag. This flag is initialized to .TRUE. when the simulation performs a refused takeoff. It causes subroutine TAKOFF to return control to the main program which subsequently calls subroutine ROLL.

SBKFLAG is the speed brake flag. This flag is set to .TRUE. after the speed brake deflection has changed.

SPLFLAG is the spoiler flag. This flag is .TRUE. when the spoiler deflection (SPOILER) is greater than 0.0; SPLFLAG is entered through namelist LND2.

SPOOL is the engine spool flag. This flag is .TRUE. when the engine spooling for a throttle transient is completed.

STEADY is the steady-state flag. This flag is set to .TRUE. during a steady-state approach when engine data is not to be determined by subroutine 'FXXENG'. Thrust and fuel flow do not change during the steady-state approach; only during the flare can thrust, fuel flow, and throttle setting be changed.

TERMFLG is the termination flag. This flag is set to .TRUE. when the any termination criterion is satisfied. This flag is currently used for takeoff and refused takeoff simulations.

VECTFLG is the thrust vectoring flag. This flag is set to .TRUE. when thrust vectoring is to be enabled for a simulation.

VFFLAG is the engine failure velocity flag. This flag is set to .TRUE. when the engine failure speed (VKFAIL) is reached.

WRITITR is the write flare iterations flag. This flag is set to .TRUE. when the user desires output on each flare iteration executed from subroutine FLARENZ. The default for WRITITR is .FALSE.; WRITITR is entered through namelist LND2.

Common CHARV

Common CHARV contains the major variables of type CHARACTER.

ENGGRP is the operating engine group. It is a user definable three-character string variable whose contents define which engines are operating during a maneuver. Examples of ENGGRP are 'AEO', 'OEI', 'IEI', and 'AEI' which correspond to all engines operating, outboard engine inoperative, inboard engine inoperative and all engines idling respectively. The default for ENGGRP is 'AEO' for all takeoffs and 'AEI' for all landings; ENGGRP is entered through namelist DATA2.

FAILGRP is the fail engine group. It is a user definable three-character string variable whose contents define the state of the engines after an engine failure or failures. The default for FAILGRP is 'OEI'; FAILGRP is entered through namelist DATA2.

FAILMOD is the engine failure mode. It is a five-character string variable whose contents define the mode of the failed engine. Usually, only two states are used: 'SEIZE' and 'SPOOL'. The default for FAILMOD is 'SEIZE'; FAILMOD is entered through namelist DATA2.

FAILST is the engine failure state. It is a four-character string variable whose contents define the state of the failed engine. Usually, only two states are used, 'IDLE' and 'OFF '. The default for FAILST is 'IDLE'; FAILST is entered through namelist DATA2.

MANUVR is the maneuver input. It is a three- or six-character string variable whose contents define which maneuver is to be simulated during execution. Valid three-character values of MANUVR are: TKO, RTO, and LND, for takeoffs, refused takeoffs, and landings respectively. Valid six-character values of MANUVR are: TOLTKO, TOLRTO, and TOLLND; for takeoffs, refused takeoffs, and landings respectively. The three-character values indicate that the program is to read MANUVR once for the entire input file. This allows the user to run one type of maneuver repetitively. For example, a single input file containing many namelist entries for landings would use MANUVR = LND. The input file would look like this:

```
LND
 &DATA DADTCMD=2.50, FLAP=30.0, GWT0= 200000., .../
 &DATA2.../
 &LND HFLARE = 20., SINKTD = 10.0,  VKAPP= 150.0 /
 &LND2.../
 &ROL IMU=1, RCR=16. /
 &ROL2 VKBRAKE=110.0 /
 &DATA DADTCMD=2.50, FLAP=30.0, GWT0= 200000., .../
 &DATA2.../
 &LND HFLARE = 15., SINKTD = 10.0,  VKAPP= 150.0 /
 &LND2.../
 &ROL IMU=1, RCR=16. /
 &ROL2 VKBRAKE=110.0 /
 &DATA DADTCMD=2.50, FLAP=30.0, GWT0= 200000., .../
 &LND HFLARE = 10, SINKTD = 10.0,  VKAPP= 150.0 /
 &LND2.../
 &ROL IMU=1, RCR=16. /
 &ROL2 VKBRAKE=110.0 /
```

The six character values indicate that the program is to read MANUVR for each set of namelist inputs within the input file. This allows the user to run different types of maneuvers in one input file. For example, a single input file containing many namelist entries for determine critical field lengths would look like this:

```
TOLTKO
 &DATA DADTCMD=2.50, FLAP=15.0, GWT= 250000., .../
 &TKO HMAX=50., THTROT=8.0, THTCLM= 8.0, VKFAIL=110.0, VKROTAT=120.0 /
 &TKO2 /
 &TKOARY /
TOLRTO
 &DATA DADTCMD=2.50, FLAP=15.0, GWT= 250000., .../
 &TKO HMAX=50., THTROT=8.0, THTCLM= 8.0, VKFAIL=110.0, VKROTAT=120.0 /
 &TKO2 /
 &TKOARY /
 &ROL IMU=1, RCR=16 /
 &ROL2 VKBRAKE=100.0 /
```

Continued takeoffs use the same MANUVR as takeoffs. There is no default for MANUVR; MANUVR is input in the main program before namelist DATA in the input file.

MVR is the first three characters of MANUVR. When MVR is set equal to 'TOL', the parameter MANUVR is read for each set of namelist inputs. (See description of MANUVR in the previous paragraphs.) There is no default for MVR; MVR is defined by the first three characters of MANUVR.

THRCRV is the thrust curve type. It is a three-character input variable that is used to describe the type of thrust curve to be utilized. This input variable allows the program to distinguish between thrust curves containing different independent (Input) variables. Examples of THRCRV are 'RC' and 'EPR'. The default for THRCRV is user defined in subroutine FORCEX; THRCRV is entered through namelist DATA2.

TKOTYPE is the takeoff type. It is a seven character input variable that is used to describe the type of takeoff as either 'STATIC' or 'ROLLING'. This input variable allows the program to distinguish between the thrust settings of either a static or rolling takeoff independently of the VKSTART input variable. The default for TKOTYPE is 'STATIC'; TKOTYPE is entered through namelist TKO2.

Common FLAPDAT

Common FLAPDAT contains the major flap variables.

IFLAP is the current element of arrays FLPARY and VFLPARY.

MAXSIZF is the maximum size of arrays FLPARY and VFLPARY. The default for MAXSIZF is 5; MAXSIZF is hard coded in subroutine TAKOFF. If MAXSIZF is increased, the dimensions for arrays FLPARY and VFLPARY must be increased accordingly.

FLPARY is the flap deflection array. This array contains five elements. This array allows the user to enter in a flap deflection schedule as a function of calibrated airspeed. At each airspeed contained in the flap deflection airspeed array (VFLPARY) the takeoff simulation changes the flap deflection (FLAP) to the corresponding flap deflection contained in the flap deflection array (FLPARY) at a rate of DFLAPDT degrees per second. The flap deflection rate (DFLAPDT) is provided by subroutine 'FXXAERO' which is a user provided subroutine called by subroutine FORCEX. If the last element of FLPARY is not 0.0, the simulation will retract the flaps to 0.0 degree at the calibrated airspeed contained in the last element of VFLPARY or at VKFLPMX which ever is less. The default for the elements of FLPARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the flap deflection until the flap retraction speed, VKFLAP, or the maximum flap deflection airspeed (VKFLPMX) is reached. The elements of FLPARY are entered through namelist TKOARY.

VFLPARY is the flap deflection airspeed array. This array contains five elements. This array allows the user to enter in a flap deflection schedule as a function of calibrated airspeed. At each airspeed contained in the flap deflection airspeed array (VFLPARY) the takeoff simulation changes the flap deflection (FLAP) to the corresponding flap deflection contained in the flap deflection array (FLPARY) at a rate of DFLAPDT degrees per second. The flap deflection rate (DFLAPDT) is provided by subroutine 'FXXAERO' which is a user provided subroutine called by subroutine FORCEX. If the last element of FLPARY is not 0.0, the simulation will retract the flaps to 0.0 degree at the calibrated airspeed contained in the last element of VFLPARY or at VKFLPMX which ever is less. The default for the elements of VFLPARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the flap deflection until the flap retraction speed, VKFLAP, or the maximum flap deflection airspeed, VKFLPMX, is reached. The elements of VFLPARY are entered through namelist TKOARY.

Common GEARDAT

Common GEARDAT contains the major landing gear variables.

IGEAR is the current element of array LGRARY.

MAXSIZG is the maximum size of array LGRARY. The default for MAXSIZG is 6; MAXSIZG is hard coded in subroutine TAKOFF. If MAXSIZG is increased, the dimension for array LGRARY must be increased accordingly.

LGRARY is the landing gear drag array. This array contains six elements. This array allows the user to enter in a landing gear drag coefficient schedule as a function of time from the start of landing gear retraction. When the landing gear retraction height (HGEAR) is reached the subroutine GRETRAC returns the gear drag coefficient increment (DCDLGR) as the value contained in the first element of LGRARY. This value remains current for one second. The value contained in the second element of the array replaces the gear drag coefficient increment previously for another 1.0 second interval. The gear drag coefficient increment (DCDLGR) must be added where the drag coefficient (CD) is summed from its components. This will usually be accomplished in the user provided subroutine 'FXXAERO'. An example table is shown below. If the last element of LGRARY is not 0.0000, the simulation will set DCDLGR to 0.0000 after drag coefficient increment The default for the elements of LGRARY are 0.0, 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation will reduce the drag coefficient linearly over the time for gear retraction (DTGEAR) in seconds. The elements of LGRARY are initialized in subroutine 'FXXAERO' which is the user provided subroutine called by subroutine FORCEX.

Array Element	C_D (without gear increment)	+	LGRARY	= C_D (with gear increment)
1	0.0280		0.0220	0.0500
2	0.0280		0.0240	0.0520
3	0.0280		0.0280	0.0560
4	0.0280		0.0160	0.0440
5	0.0280		0.0070	0.0350
6	0.0280		0.0000	0.0280

Notice the total landing gear increment is removed between the 5th and 6th second after the initiation of landing gear retraction.

Common VECTDAT

Common VECTDAT contains the major thrust vectoring variables.

IVECT is the current element of arrays XNUARY, HVCTARY, and VVCTARY.

MAXSIZV is the maximum size of arrays XNUARY, HVCTARY, and VVCTARY. The default for MAXSIZV is 5; MAXSIZV is hard coded in subroutine TAKOFF. If MAXSIZV is increased, the dimensions for arrays XNUARY, HVCTARY, and VVCTARY must be increased accordingly.

XNUARY is the vectored thrust angle array. This array contains five elements. At each altitude contained in the vectored thrust altitude array (HVCTARY) or at each calibrated airspeed contained in the vectored thrust airspeed array (VVCTARY) the takeoff simulation changes the vectored thrust angle (VTANGLE) to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is a user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of VVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle. The elements of XNUARY are entered through namelist TKOARY.

HVCTARY is the vectored thrust altitude array. This array contains five elements. At each altitude contained in the vectored thrust altitude array (HVCTARY) the takeoff simulation changes the vectored thrust angle (VTANGLE) to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is a user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of HVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle. The elements of HVCTARY are entered through namelist TKOARY.

VVCTARY is the vectored thrust airspeed array. This array contains five elements. At each calibrated airspeed contained in the vectored thrust airspeed array (VVCTARY) the takeoff simulation changes the vectored thrust angle (VTANGLE) to the corresponding vectored thrust angle contained in the vectored thrust angle array (XNUARY) at a rate of DVECTDT degrees per second. The vectored thrust angle rate (DVECTDT) is provided by subroutine 'FXXENG' which is a user provided subroutine called by subroutine FORCEX. If both the HVCTARY and VVCTARY arrays are used, the HVCTARY array takes precedence. The default for the elements of VVCTARY are 0.0, 0.0, 0.0, 0.0, 0.0. At this default setting, the takeoff simulation does not change the vectored thrust angle. The elements of VVCTARY are entered through namelist TKOARY.

Common CINDEX

Common CINDEX contains major random access curve file constants.

MSTRNDX is the random access curves master index. It is an array of NDXLEN (3), integers initialized by subroutine OPENMS, a CDC Cyber specific FORTRAN subroutine.

NDXLEN is the length of the master index. NDXLEN is an integer constant equal to three; NDXLEN is initialized in subroutine INICURV.

LUCURV is the logical unit for the curve file. It is an integer variable which determines the local file name of the random access curve file. The default for LUCURV is 2 (which corresponds to file TAPE2); LUCURV is initialized in subroutine INICURV. Currently, no other value is valid for this parameter.

Common RACURV

Common RACURV contains the major random access curve file control variables as well as the random access curve file names.

IDBUG is the UFTAS debug flag. It can provide debug information concerning the UFTAS random access curve file subroutines. The default for IDBUG is 0; IDBUG is entered through namelist DATA2.

IEXTOR is the extrapolation override code. When set to a non-zero value, IEXTOR overrides the curve extrapolation code. This flag is intended to be used in only specialized cases. The default for IEXTOR is 0; IEXTOR is initialized in subroutine INICURV.

KURNAM is the random access curve file name array. It contains the names of up to 99 random access curve file names as Hollerith strings stored in an array of type INTEGER.

Common VALUES

Common VALUES contains the arrays which contain the values passed to and from the random access curve files. Examples for array names are VALIN, VAERO, and VENGINE for the input values array, the aerodynamic values output array, and the engine values output array respectively. This common block is user defined and largely determined by the curve file structure.

Common TABLES

Common TABLES contains the table arrays used to store raw values from the random access curve files. The values which are passed to the output values arrays are interpolated from these raw values. Examples for array names are TAERO and TENGINE for the aerodynamic output table array and the engine output table array respectively. This common block is user defined and determined by the curve file structure. The size of each of the table arrays is provided to the user when the random access curve file is generated.

REFERENCES

1. Bowles, Jeff V., and Galloway, Thomas L., *Computer Programs for Estimating Aircraft Takeoff and Landing Performance*, NASA Technical Memorandum X.62,333, 1973.

2. Olson, Wayne, NASA Ames Takeoff and Landing Simulation Computer Program Supplementary Documentation, circa 1979.

3. MIL-C-005011B (USAF), *Charts: Standard Aircraft Characteristics And Performance, Pilot Aircraft (Fixed Wing)*, 21 June 1977.

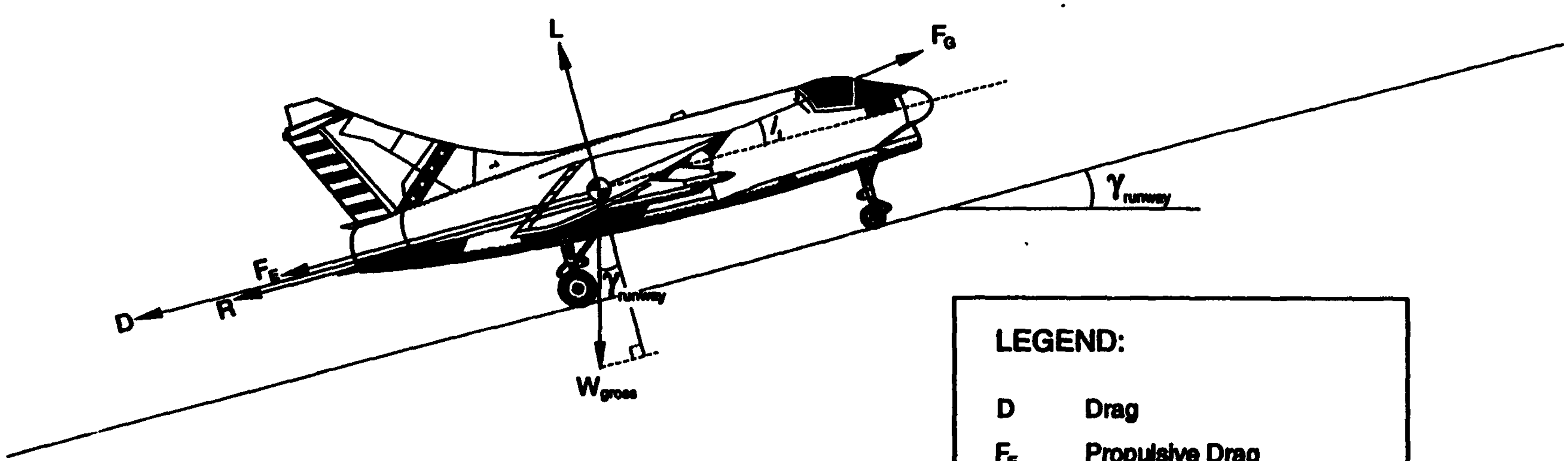
4. Morris, John L.L., *Computational Methods in Elementary Numerical Analysis*, John Wiley and Sons, LTD, 1983, ISBN 0 471 10419 1.

BIBLIOGRAPHY

1. Batt, James T., F-15C Limited Takeoff and Landing Evaluation, AFFTC-TR-81-18, Air Force Flight Test Center, Edwards AFB, California, January 1982.

2. Jabour, William J.; Jones, Lyle W.; and Friedman, Lee M., F-16A Braking Evaluation, AFFTC-TR-85-05, Air Force Flight Test Center, Edwards AFB, California, June 1985.

FREE - BODY DIAGRAM



$$R = \mu[W_{gross} \cos \gamma_{runway} - L - F_G \sin(\alpha + l_t)]$$

$$\frac{dV_t}{dt} = \frac{g}{W_{gross}} [F_G \cos(\alpha + l_t) - F_E - D - R - W_{gross} \sin \gamma_{runway}]$$

ASSUMPTIONS:

-- Wings Level
-- Zero Sideslip
-- Point Mass
-- Three-point Attitude

LEGEND:

D	Drag
F_E	Propulsive Drag
F_G	Gross Thrust
g	Acceleration due to gravity
l_t	Thrust Incidence Angle
L	Lift
R	Runway Friction
t	time
V_t	True Airspeed
W_{gross}	Gross Weight
α	Angle of Attack
γ_{runway}	Runway Slope
μ	Coefficient of Friction

APPENDIX A: EQUATIONS OF MOTION

This section shows the equations of motion used in Subroutines FORCEX, DERIVGR, DERIVAT, and DERIVAL. The aircraft is treated as a point mass confined to motion in a vertical plane, and the rotational dynamics have been neglected. This simplification requires the input of the angular rates and precludes the use of this program as a tool to simulate minimum control speeds. These rates are approximated by a finite difference form or input by the user as commanded rates.

Force coefficient equations:

$$C_X = C_D - [F_G(\cos(\alpha + i_t) - F_E]/qS_{wing} \qquad \text{(positive aft)} \qquad (1)$$

$$C_Y = C_L + [F_G(\sin(\alpha + i_t)\quad]/qS_{wing} \qquad \text{(positive up)} \qquad (2)$$

Equation of motion during ground roll:

$$dV_t/dt = (g/W_{gross})\{-W_{gross}[\sin(\gamma_{runway}) + \mu_{runway}\cos(\gamma_{runway})] + qS_{wing}(C_Y\mu_{runway} - C_X)\} \qquad (3)$$

Equation of motion along flightpath:

$$dV_t/dt = (g/W_{gross})\{-W_{gross}\sin(\gamma) - qS_{wing}(C_X)\} \qquad (4)$$

Equation of motion normal to flightpath:

$$V_t(d\gamma/dt) = (g/W_{gross})\{-W_{gross}\cos(\gamma) + qS_{wing}(C_Y)\} \qquad (5)$$

Load factor equation:

$$n_z = qS_{wing}C_Y/(W_{gross}) + [1.0 - \cos(\gamma)] \qquad (6)$$

Rate of Climb equation:

$$R/C = V_t\sin(\gamma) \qquad (7)$$

where:

C_D = aerodynamic drag coefficient
C_L = aerodynamic lift coefficient
C_X = force coefficient along flightpath
C_Y = force coefficient normal to flightpath
F_E = ram drag
F_G = gross thrust
dt = delta time
g = acceleration due to gravity
n_z = load factor
q = dynamic pressure
R/C = rate of climb
S_{wing} = reference wing area
V_t = true airspeed
W_{gross} = gross weight
α = angle of attack
γ = flightpath angle
γ_{runway} = runway slope
μ_{runway} = runway friction coefficient (either braking or rolling)

Rotational rate approximations by finite difference form

$$\theta = \gamma + \alpha \tag{8}$$

Differentiating with respect to time

$$d\theta/dt = d\gamma/dt + d\alpha/dt \tag{9}$$

Divide equation 5 by V_t

$$d\gamma/dt = [g/(V_t W_{gross})] \{-W_{gross}\cos(\gamma) + qS_{wing}(C_Y)\} \tag{10}$$

Finite difference approximation

$$d\alpha/dt \cong (\alpha - \alpha_j)/\Delta t \tag{11}$$

Substituting equations 10 and 11 into equation 9

$$d\theta/dt = [g/(V_t W_{gross})] \{-W_{gross}\cos(\gamma) + qS_{wing}(C_Y)\} + (\alpha - \alpha_j)/\Delta t \tag{12}$$

where:

- dt = delta time
- Δt = integration step size
- α = current angle of attack
- α_j = previous angle of attack
- γ = flightpath angle
- θ = pitch attitude

Landing equations

The flare height can be defined as a function of the flare radius (R) and the flare radius can be represented using the circular centripetal acceleration equation. For constant normal load facto (n_z) and constant airspeed ($V_{t_{app}}$)

$$h_{flare} = Y_{TD} - Y_{app} = R\cos\gamma_{TD} - R\cos\gamma_{app} = R(\cos\gamma_{TD} - \cos\gamma_{app}) \tag{13}$$

$$a_n = V_{t_{app}}^2/R = g(n_z - 1.0) \tag{14}$$

Substituting equation 14 for R into equation 13,

$$h_{flare} = [V_{t_{app}}^{2}(\cos\gamma_{TD} - \cos\gamma_{app})/g]/(n_z - 1.0) \tag{15}$$

Solving for normal load factor as a function of the two flightpath angles and approach speed,

$$n_z = [V_{t_{app}}^{2}(\cos\gamma_{TD} - \cos\gamma_{app})/g\ h_{flare}] + 1.0 \tag{16}$$

where:

a_n = acceleration normal to flightpath
g = acceleration due to gravity
h_{flare} = flare initiation altitude (AGL)
n_z = normal load factor
R = flare radius
$V_{t_{app}}$ = approach true airspeed
γ_{app} = approach flightpath angle
γ_{TD} = touchdown flightpath angle

This gives subroutine FLARENZ a starting point for iterating on normal load factor to match the sink rate at touchdown.

Using the following equation for kinetic energy, brake energy is calculated with equation 18;

$$E_{kinetic} = ½\,(W_{gross}/g)\ V_{t_g}^{\ 2} \tag{17}$$

$$E_{brake} = E_{kinetic} + \Sigma(F_N - C_D q S_{wing})\ V_{t_g}\,\Delta t \tag{18}$$

where:

C_D = aerodynamic drag coefficient
F_N = net thrust
E_{brake} = brake energy
$E_{kinetic}$ = kinetic energy
g = acceleration due to gravity
q = dynamic pressure
S_{wing} = reference wing area
V_{t_g} = true ground speed
W_{gross} = gross weight
Δt = delta time

Main Gear Normal Force equations during braking

The normal force on the main gear can be determined by summing the forces in both the x- and y-axes and summing the moments about the center of gravity. The rolling coefficient of friction is applied to the nose gear normal force and the braking coefficient of friction is applied to the main gear normal force. The following equations assume no aerodynamic loading of the tail, no moments from spoilers or other control surfaces, and the center of gravity is located forward of the braking gear (the main).

Summing the forces in the x-axis,

$$-(W_{gross}/g)\, dV_t/dt = C_X\,(q\, S_{wing}) + W_{gross}\sin\gamma_{runway} + \mu_{rolling}\, N_{nose} + \mu_{brake}\, N_{brake} \qquad (19)$$

Summing the forces in the y-axis,

$$0 = C_Y\,(q\, S_{wing}) - W_{gross}\cos\gamma_{runway} + N_{nose} + N_{main} \qquad (20)$$

Summing the moments about the center of gravity,

$$0 = N_{nose}\, X_{nose} - N_{main}\, X_{main} - \mu_{rolling}\, N_{nose}\, Y_{cg} - \mu_{brake}\, N_{main}\, Y_{cg} \qquad (21)$$

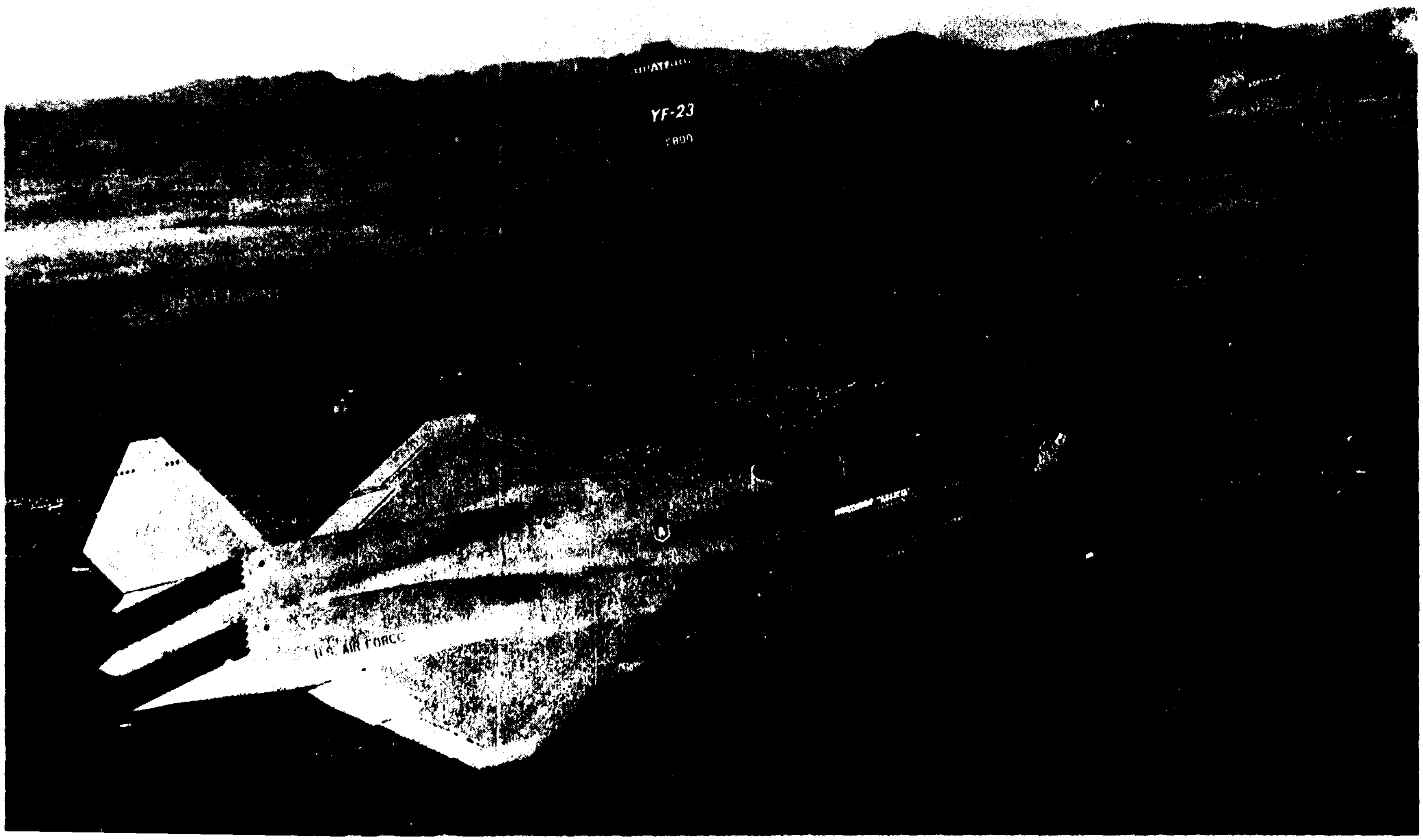
Solving for N_{main},

$$N_{main} = \frac{[(W_{gross}/g)dV_t/dt + C_X(qS_{wing}) + W_{gross}\sin\gamma_{runway}]Y_{cg} + [W_{gross}\cos\gamma_{runway} - C_Y(qS_{wing})]X_{nose}}{(X_{main} + X_{nose})} \qquad (22)$$

where:

C_X = force coefficient along flightpath
C_Y = force coefficient normal to flightpath
dV_t/dt = acceleration
g = acceleration due to gravity
N_{main} = main gear normal force
N_{nose} = nose gear normal force
q = dynamic pressure
S_{wing} = reference wing area
W_{gross} = gross weight
X_{main} = horizontal distance from main gear to center of gravity
X_{nose} = horizontal distance from nose gear to center of gravity
Y_{cg} = vertical distance from ground to center of gravity
γ_{runway} = runway slope
μ_{brake} = braking friction coefficient
$\mu_{rolling}$ = rolling friction coefficient

Distances are all positive values. Any units for distance, if applied consistently, will work in these equations.


YF-23
U.S. AIR FORCE

APPENDIX B: SOURCE CODE LISTING

This section provides the user with a complete source code listing of the program and calling subroutines. Curve file subroutines are not included.

Main Program

```
      PROGRAM TOLAND                                                     TOLAND   1
C****                                                                    TOLAND   2
C**** SUBROUTINES-   INITIAL      SPOIL      FLARE      SPEED      INTG  TOLAND   3
C****                 PITCH     TVECTOR       ROLL    DERIVGR   STEDYST  TOLAND   4
C****               FRETRAC      TAKOFF      ERROR    DERIVAT   FLARENZ  TOLAND   5
C****               GRETRAC      LANDNG       HALT    DERIVAL    ITRLND  TOLAND   6
C****               SPDBRAK     APPROCH    ATMOSPH       INTX            TOLAND   7
C****                                                                    TOLAND   8
C**** FUNCTIONS-       DGDT       DVTDH    DDELTDH     INTERP     ZEROX  TOLAND   9
C****                  DVDT        DADH     DSIGDH                       TOLAND  10
C****                                                                    TOLAND  11
C**** USER PROVIDED SUBROUTINES-                                         TOLAND  12
C****               INICURV   'FXXAERO'   'FXXENG'   SPOOLDNF     GENMU  TOLAND  13
C****                FORCEX     GEFFECT    SPOOLUP   SPOOLDNR            TOLAND  14
C****                                                                    TOLAND  15
C**** TAKEOFF, LANDING AND REFUSED TAKEOFF SIMULATION                    TOLAND  16
C****                                                                    TOLAND  17
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                    CTRL   1
      PARAMETER (LUIN=3,LUOUT=4)                                           CTRL   2
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL   3
     &                 NEQ,NPAGE,TIME,TIMEROL                              CTRL   4
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,  AIRCRFT  1
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                 AIRCRFT  2
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,       AERO   1
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                            AERO   2
      COMMON/ENGINE/   AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,      ENGINE   1
     &                 REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,        ENGINE   2
     &                 XENGOUT,XIDLE,XMIL,ZFN                              ENGINE   3
      COMMON/VECTOR/   HVECT,VVECT                                         VECTOR   1
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ  AIRBORN   1
      COMMON/AIRSPED/  VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,  AIRSPED   1
     &                 VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                 AIRSPED   2
      COMMON/RUNWAY/   ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,      RUNWAY   1
     &                 HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD, RUNWAY   2
     &                 TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                 RUNWAY   3
      DOUBLEPRECISION  VTAS,DIST,ACCEL,VTASJ,RKGRND                         INTEG   1
      COMMON/INTEG/    VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                     INTEG   2
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,   FPINTEG   1
     &                 ROC,RKAIR                                          FPINTEG   2
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,   FPINTEG   3
     &                 ROC,RKAIR(40)                                      FPINTEG   4
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF  ATMOS   1
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                      CONST   1
     &            RX=57.29577951308,TSLF=59.0,ZERO=0.0)                     CONST   2
      COMMON/CONST/    ASLSQR5,TWOOVR7                                      CONST   3
```

```
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,          FLAGS      1
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,            FLAGS      2
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,            FLAGS      3
     &         VFFLAG,WRITITR                                                   FLAGS      4
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,          FLAGS      5
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,          FLAGS      6
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,      FLAGS      7
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                           FLAGS      8
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,           CHARV      1
     &          THRCRV*3,TKOTYPE*7                                              CHARV      2
      COMMON/CHARV/    ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,         CHARV      3
     &                 TKOTYPE                                                  CHARV      4
      COMMON/FLAPDAT/  IFLAP,MAXSIZF,FLPARY(5),VFLPARY(5)                       FLAPDAT    1
      REAL LGRARY(6)                                                            GEARDAT    1
      COMMON/GEARDAT/  IGEAR,MAXSIZG,LGRARY                                     GEARDAT    2
      COMMON/VECTDAT/  IVECT,MAXSIZV,XNUARY(5),HVCTARY(5),VVCTARY(5)            VECTDAT    1
      COMMON/CINDEX/   MSTRNDX(3),NDXLEN,LUCURV                                 CINDEX     1
      COMMON/RACURV/   IDBUG,IEXTOR,KURNAM(99)                                  RACURV     1
      LOGICAL MOREFLG                                                           TOLAND    36
      NAMELIST/DATA/CGPCT,DADTCMD,DTEMPF,FLAP,GAMMARW,GWT0,HCLEAR,              TOLAND    37
     &              HRUNWAY,VKWIND                                              TOLAND    38
      NAMELIST/DATA2/AOA3PT,CONFIG,DCDX,DCLX,DTIME,DTFAIL,ENGGRP,EPR,           TOLAND    39
     &              FAILGRP,FAILMOD,FAILST,FLPPCT,FLT,FLTNDX,FGPCT,             TOLAND    40
     &              IDBUG,JDEBUG,LOADING,LUMSG,PWRCODE,RC,REVFLAG,              TOLAND    41
     &              ROLLMU,SPDBRK0,THRCRV,VKMCG,XENG,XENGFLD                    TOLAND    42
C****                                                                           TOLAND    43
C**** OPEN INPUT AND OUTPUT FILE                                                TOLAND    44
      OPEN (UNIT=LUIN,FILE=' ')                                                 TOLAND    45
      OPEN (UNIT=LUOUT,FILE=' ')                                                TOLAND    46
C****                                                                           TOLAND    47
C**** MANUVR  TKO,TOLTKO    TAKEOFF                                             TOLAND    48
C****         LND,TOLLND    LANDING                                             TOLAND    49
C****         RTO,TOLRTO    REFUSED TAKEOFF                                     TOLAND    50
   15 READ(LUIN,1001,END=99)MANUVR                                              TOLAND    51
 1001 FORMAT(A6)                                                                TOLAND    52
C****                                                                           TOLAND    53
C**** RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY), THEN          TOLAND    54
C**** READ NAMELIST DATA FROM INPUT FILE (LUIN).  PARAMETERS NOT RESET          TOLAND    55
C**** HERE RETAIN THEIR VALUE FOR THE SUBSEQUENT RUN IF NOT EXPLICITLY          TOLAND    56
C**** DECLARED IN NAMELIST DATA.                                                TOLAND    57
   25 CALL INITIAL('ZERO',ALPHA,FLAP0,GWT0,SPDBRK0)                             TOLAND    58
      READ(LUIN,DATA,END=99)                                                    TOLAND    59
      READ(LUIN,DATA2)                                                          TOLAND    60
      CALL INITIAL('MAIN',ALPHA,FLAP0,GWT0,SPDBRK0)                             TOLAND    61
      IF(MOREFLG) WRITE(LUOUT,2001)                                             TOLAND    61
 2001 FORMAT('1')                                                               TOLAND    62
```

```
      IF   (MANUVR.EQ.'TKO  '.OR.MANUVR.EQ.'TOLTKO') THEN                    TOLAND
C****    TAKEOFF ROUTINE                                                     TOLAND
         CALL TAKOFF(ALPHA,GDIST)                                            TOLAND
         WRITE(LUOUT,2002) FLT,GWT0,DCDX,FGPCT,VKMCG,FLAP0,DTEMPF,DCLX,      TOLAND
     &                     FAILST,ENGGRP                                     TOLAND
2002     FORMAT( ' FLT =',F8.4,' WT =',F8.0,' DCD =',F7.4,                   TOLAND
     &           ' FGPCT =',F7.1,' VKMCG =',F7.1,' FLAP =',F6.1,/,          TOLAND
     &           ' DTEMP =',F7.1,' DEGREES F ',' DCL =',F7.4,               TOLAND
     &           ' FAILST =',A5,'; ENGGRP =',A4,/)                           TOLAND
      ELSEIF(MANUVR.EQ.'RTO  '.OR.MANUVR.EQ.'TOLRTO') THEN                   TOLAND
C****    REFUSED TAKEOFF ROUTINE                                             TOLAND
         CALL TAKOFF(ALPHA,GDIST)                                            TOLAND
         CALL ROLL(ALPHA,GDIST)                                              TOLAND
         WRITE(LUOUT,2002) FLT,GWT0,DCDX,FGPCT,VKMCG,FLAP0,DTEMPF,DCLX,      TOLAND
     &                     FAILST,ENGGRP                                     TOLAND
      ELSEIF(MANUVR.EQ.'LND  '.OR.MANUVR.EQ.'TOLLND') THEN                   TOLAND
C****    LANDING ROUTINE                                                     TOLAND
         CALL LANDNG(ALPHA)                                                  TOLAND
         WRITE(LUOUT,2003) FLT,DCDX,DTEMPF,BRKFCTR,DCLX,FLAP0                TOLAND
2003     FORMAT(' FLT =',F12.4,' DCD =',F7.4,' DTEMP =',F6.1,' DEGREES',     TOLAND
     &          'F',/,' BRKFCTR =',F8.3,' DCL =',F7.4,' FLAP  =',F6.1,/)     TOLAND
      ENDIF                                                                  TOLAND
      MOREFLG = .TRUE.                                                       TOLAND
      IF(MVR.EQ.'TOL') THEN                                                  TOLAND
         GO TO 15                                                            TOLAND
      ELSE                                                                   TOLAND
         GO TO 25                                                            TOLAND
      ENDIF                                                                  TOLAND
   99 CALL HALT(LUIN,LUMSG,LUOUT,' TOLAND NORMAL TERMINATION')               TOLAND
      END                                                                    TOLAND
```

Program Subroutines

Subroutine INITIAL

```
      SUBROUTINE INITIAL(GROUP,ALPHA,FLAP0,GWT0,SPDBRK0)                        INITIAL
C**** THIS SUBROUTINE INITIALIZES PROGRAM VARIABLES.                            INITIAL
C****                                                                           INITIAL
      CHARACTER*4 GROUP                                                         INITIAL
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                         CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                                CTRL
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,     CTRL
     &                NEQ,NPAGE,TIME,TIMEROL                                    CTRL
      COMMON/AIRCRFT/ AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,        AIRCRFT
     &                LOADING,SWING,THTMAX,WNGLOD,XLFMAX                        AIRCRFT
      COMMON/AERO/    CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,           AERO
     &                SPDBRK,SPOILER,VKCAS,VKTAS                                AERO
      COMMON/ENGINE/  AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,            ENGINE
     &                REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,              ENGINE
     &                XENGOUT,XIDLE,XMIL,ZFN                                    ENGINE
      COMMON/VECTOR/  HVECT,VVECT                                               VECTOR
      COMMON/AIRBORN/ ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ         AIRBORN
      COMMON/AIRSPED/ VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,         AIRSPED
     &                VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                        AIRSPED
      COMMON/RUNWAY/  ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,            RUNWAY
     &                HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,       RUNWAY
     &                TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                       RUNWAY
      DOUBLEPRECISION VTAS,DIST,ACCEL,VTASJ,RKGRND                              INTEG
      COMMON/INTEG/   VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                          INTEG
      DOUBLEPRECISION FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,          FPINTEG
     &                ROC,RKAIR                                                 FPINTEG
      COMMON/FPINTEG/ FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,          FPINTEG
     &                ROC,RKAIR(40)                                             FPINTEG
      COMMON/ATMOS/   TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF     ATMOS
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                          CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                          CONST
      COMMON/CONST/   ASLSQR5,TWOOVR7                                           CONST
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,          FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,            FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,            FLAGS
     &         VFFLAG,WRITITR                                                   FLAGS
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,           FLAGS
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,           FLAGS
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,       FLAGS
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                            FLAGS
      CHARACTER  ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,          CHARV
     &           THRCRV*3,TKOTYPE*7                                             CHARV
      COMMON/CHARV/   ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,          CHARV
     &                TKOTYPE                                                   CHARV
      COMMON/FLAPDAT/ IFLAP,MAXSIZF,FLPARY(5),VFLPARY(5)                        FLAPDAT
      REAL LGRARY(6)                                                            GEARDAT
      COMMON/GEARDAT/ IGEAR,MAXSIZG,LGRARY                                      GEARDAT
      COMMON/VECTDAT/ IVECT,MAXSIZV,XNUARY(5),HVCTARY(5),VVCTARY(5)             VECTDAT
```

```
      COMMON/RACURV/   IDBUG,IEXTOR,KURNAM(99)                                    RACURV    1
      LOGICAL MSGFLAG                                                             INITIAL  23
      DATA  DADTCMD, DCDX, DCLX, DTDTGEX, DTEMPF, DTIME, EPR, FAILGRP, FAILMOD,    INITIAL  24
     &     /       2.5,    0.0,    0.0,       1.0,      0.0, 0.10D0,  0.0,    'OEI',   'SEIZE', /,   INITIAL  25
     &        FAILST, GAMMARW, HRUNWAY, HCLEAR, ICOUNT, LUMSG,NPAGE, ROLLMU       INITIAL  26
     &     /   'IDLE',        0.0,        0.0,     50.0,      1,      0,    58,    0.025 /   INITIAL  27
     &        VKMCG                                                               INITIAL  27
     &     /      0.0 /                                                           INITIAL  28
      DATA FLPARY   /0.0,0.0,0.0,0.0,0.0,0.0/,VFLPARY/ 0.0, 0.0, 0.0, 0.0, 0.0/   INITIAL  30
     &        ,HVCTARY /0.0,0.0,0.0,0.0,0.0,0.0/,VVCTARY/999.,999.,999.,999.,999./   INITIAL  31
     &        ,XNUARY  /0.0,0.0,0.0,0.0,0.0,0.0/,MAXSIZF,MAXSIZG,MAXSIZV/5,6,5/   INITIAL  32
      IF  (GROUP.EQ.'ZERO') THEN                                                  INITIAL  33
C****        INITIALIZE CONSTANTS.                                                INITIAL  34
C****        ASLSQR5  = 661.48*SQRT(5.0); TWOOVR7 = 2.0/7.0                       INITIAL  35
             ASLSQR5  = ASL*SQRT(5.0)                                             INITIAL  36
             TWOOVR7 = 2.0/7.0                                                    INITIAL  37
C****                                                                             INITIAL  38
C****        INITIALIZE MVR WITH THE FIRST THREE CHARACTERS OF MANUVR.  THIS      INITIAL  39
C****        ALLOWS THE PROGRAM TO DISTINGUISH IF THE INPUT FILE CONTAINS ALL     INITIAL  40
C****        ONE TYPE OF MANEUVERS (I.E. RTOS) OR A VARIETY.  IF THE INPUT        INITIAL  41
C****        FILE CONTAINS A VARIETY, THE PROGRAM WILL LOOP TO LINE 15 TO READ    INITIAL  42
C****        THE NEW MANUVR INSTEAD OF LINE 25.                                   INITIAL  43
             MVR = MANUVR(1:3)                                                    INITIAL  44
C****                                                                             INITIAL  45
C****        RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY),          INITIAL  46
C****        BEFORE READING NAMELIST DATA FROM INPUT FILE (TAPE5).                INITIAL  47
C****        PARAMETERS NOT RESET HERE RETAIN THEIR VALUE FOR THE SUBSEQUENT      INITIAL  48
C****        RUN IF NOT EXPLICITLY DECLARED IN NAMELIST DATA.                     INITIAL  49
             AOA3PT   = ZERO                                                      INITIAL  50
             DTFAIL   = ZERO                                                      INITIAL  51
             ENGGRP   = 'NUL'                                                     INITIAL  52
             FLAP     = -1.0                                                      INITIAL  53
             FPCTFLG  = .FALSE.                                                   INITIAL  54
             GWT0     = ZERO                                                      INITIAL  55
             IDBUG    = 0                                                         INITIAL  56
             JDEBUG   = 0                                                         INITIAL  57
             REVFLAG  = .FALSE.                                                   INITIAL  58
             TIMEROL  = ZERO                                                      INITIAL  59
             VKWIND   = ZERO                                                      INITIAL  60
             XENG     = ZERO                                                      INITIAL  61
             XENGFLD  = 1.0                                                       INITIAL  62
```

```
      ELSE
C****      INITIALIZE LUMSG.
           IF         (LUMSG.EQ.0 .OR. LUMSG.EQ.LUIN)          THEN
               LUMSG = LUOUT
           ELSEIF     (LUMSG.NE.LUOUT .AND. .NOT.(MSGFLAG)) THEN
               MSGFLAG = .TRUE.
               OPEN(UNIT=LUMSG,FILE='TOLAND.MSG')
           ENDIF
C****      INITIALIZE PROGRAM CONTROL PARAMETERS.
           DADTCMD = ABS(DADTCMD)
           DTIMEJ    = DTIME
           KENG      = 0
           LINENUM   = 6
           FLAPFLG   = .TRUE.
           LGRFLAG   = .TRUE.
           VECTFLG   = .TRUE.
           BRKFLAG   = .FALSE.
           CLRHGT    = .FALSE.
           ERRFLAG   = .FALSE.
           SBKFLAG   = .FALSE.
           SPLFLAG   = .FALSE.
           TERMFLG   = .FALSE.
           VFFLAG    = .FALSE.
C****
C****      INITIALIZE AIRCRAFT CONSTANTS.
           FLAP0     = FLAP
           GRW       = GAMMARW/RX
           GWT       = GWT0
           VWIND     = VKWIND*FPSKTS
           WNGLOD    = GWT0/SWING
           XMU       = ROLLMU
C****
C****      TERMINATE PROGRAM IF GROSS WEIGHT NOT INITIALIZED.
           IF(GWT0.EQ.ZERO) CALL HALT(LUIN,LUMSG,LUOUT,
     &                           ' GROSS WEIGHT NOT INPUT BY USER.')
C****
C****      INITIALIZE THE NUMBER OF ENGINES BEFORE AND AFTER ENGINE
C****      FAILURE.
           IF (NENG.EQ.    0)  NENG = 1
           IF (XENG.EQ.ZERO) XENG = FLOAT(NENG)
           IF (FAILST.EQ.'MIL ') THEN
               XENGFLD = 1.0 - XMIL
           ENDIF
           XENGOUT = XENG - XENGFLD
C****
C****      INITIALIZE LOGIC CONTROL VARIABLES FOR FLAPS, GEAR AND THRUST
C****      VECTORING.
           IFLAP = IGEAR   = IVECT = 1
C****
C****      INITIALIZE INTEGRATION VARIABLES.
           RKGRND(14)  = ZERO
           RKAIR   (27)  = ZERO
```

```
      IF   (MANUVR.EQ.'TKO   '.OR.MANUVR.EQ.'TOLTKO') THEN               INITIAL
C****      INITIALIZE FLAGS FOR THE TAKEOFF MANEUVERS.                   INITIAL
           AOA0FLG  = .TRUE.                                             INITIAL
           GEFLAG   = .TRUE.                                             INITIAL
           LIFTOFF  = .FALSE.                                            INITIAL
           ROTATE   = .FALSE.                                            INITIAL
           RTOFLAG  = .FALSE.                                            INITIAL
           IF(ENGGRP.EQ.'NUL') ENGGRP = 'AEO'                            INITIAL
C****                                                                    INITIAL
C****      RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY).   INITIAL
C****      PARAMETERS NOT RESET HERE RETAIN THEIR VALUE FOR THE          INITIAL
C****      SUBSEQUENT RUN IF NOT EXPLICITLY DECLARED IN NAMELIST TKO.    INITIAL
           TKOTYPE  = 'STATIC'                                           INITIAL
           NEQ      = 2                                                  INITIAL
           ALPHA    = ZERO                                               INITIAL
           DIST     = ZERO                                               INITIAL
           HAGL     = ZERO                                               INITIAL
           PRESALT  = HRUNWAY                                            INITIAL
           THETAF   = AOA3PT                                             INITIAL
           TIME     = ZERO                                               INITIAL
           VKFAIL   = ZERO                                               INITIAL
           VKROTAT  = ZERO                                               INITIAL
           VKSTART  = ZERO                                               INITIAL
           XLF      = 1.0                                                INITIAL
      ELSEIF(MANUVR.EQ.'RTO   '.OR.MANUVR.EQ.'TOLRTO') THEN              INITIAL
C****      INITIALIZE FLAGS FOR THE REFUSED TAKEOFF MANEUVERS.           INITIAL
           AOA0FLG  = .TRUE.                                             INITIAL
           GEFLAG   = .TRUE.                                             INITIAL
           RTOFLAG  = .TRUE.                                             INITIAL
           LIFTOFF  = .FALSE.                                            INITIAL
           ROTATE   = .FALSE.                                            INITIAL
           IF(ENGGRP.EQ.'NUL') ENGGRP = 'AEO'                            INITIAL
C****                                                                    INITIAL
C****      RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY).   INITIAL
C****      PARAMETERS NOT RESET HERE RETAIN THEIR VALUE FOR THE          INITIAL
C****      SUBSEQUENT RUN IF NOT EXPLICITLY DECLARED IN NAMELIST TKO.    INITIAL
           TKOTYPE  = 'STATIC'                                           INITIAL
           NEQ      = 2                                                  INITIAL
           ALPHA    = ZERO                                               INITIAL
           DIST     = ZERO                                               INITIAL
           HAGL     = ZERO                                               INITIAL
           PRESALT  = HRUNWAY                                            INITIAL
           THETAF   = ZERO                                               INITIAL
           TIME     = ZERO                                               INITIAL
           VKFAIL   = ZERO                                               INITIAL
           VKROTAT  = ZERO                                               INITIAL
           VKSTART  = ZERO                                               INITIAL
           XLF      = 1.0                                                INITIAL
```

```
      ELSEIF(MANUVR.EQ.'LND   '.OR.MANUVR.EQ.'TOLLND') THEN      INITIAL
C****      INITIALIZE FLAGS FOR THE LANDING MANEUVER.            INITIAL
           AOA0FLG = .FALSE.                                     INITIAL
           GEFLAG  = .FALSE.                                     INITIAL
           RTOFLAG = .FALSE.                                     INITIAL
           LIFTOFF = .TRUE.                                      INITIAL
           ROTATE  = .TRUE.                                      INITIAL
           IF(ENGGRP.EQ.'NUL') ENGGRP = 'AEI'                    INITIAL
C****                                                            INITIAL
C****      RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY).  INITIAL
C****      PARAMETERS NOT RESET HERE RETAIN THEIR VALUE FOR THE  INITIAL
C****      SUBSEQUENT RUN IF NOT EXPLICITLY DECLARED IN NAMELIST LND.  INITIAL
           NEQ     = 4                                           INITIAL
           ALPHA   =99.0                                         INITIAL
           GAMMAPP=  1.0                                         INITIAL
           HFLARE  = -1.0                                        INITIAL
           STEADY  = .TRUE.                                      INITIAL
           VKAPP   =  1.0                                        INITIAL
           WRITITR = .FALSE.                                     INITIAL
C****                                                            INITIAL
C****      INITIALIZE ADDITIONAL PARAMETERS.                     INITIAL
           ALPHMX  = THTMAX                                      INITIAL
           HAGL    = HCLEAR                                      INITIAL
           IF(VKFLAP.EQ.ZERO) VKFLAP = VKFLPMX                   INITIAL
      ENDIF                                                      INITIAL
      CALL SPDBRAK ('RESET',SPDBRK,SPDBRK0,ZERO)                 INITIAL
      CALL SPOIL   ('RESET',SPOILER,ZERO   ,ZERO)                INITIAL
   ENDIF                                                         INITIAL
   RETURN                                                        INITIAL
   END                                                           INITIAL
```

8Subroutine FRETRAC

```
      SUBROUTINE FRETRAC(FLAP,VKFLAP,DFLAPDT)                          FRETRAC
C**** THIS SUBROUTINE CONTROLS RETRACTION OF THE FLAPS BASED UPON FLAP FRETRAC
C**** RETRACTION SPEED (VKFLAP) AND THE DELTA TIME FOR FLAP RETRACTION FRETRAC
C**** (DFLAPDT).                                                       FRETRAC
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                       CTRL
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL
     &                NEQ,NPAGE,TIME,TIMEROL                           CTRL
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,  FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,    FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,    FLAGS
     &         VFFLAG,WRITITR                                          FLAGS
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,   FLAGS
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,   FLAGS
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,  FLAGS
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                   FLAGS
      COMMON/FLAPDAT/ IFLAP,MAXSIZF,FLPARY(5),VFLPARY(5)               FLAPDAT
      DOUBLE PRECISION TIMEJ                                           FRETRAC
      LOGICAL INIFLAP                                                  FRETRAC
      IF(.NOT.INIFLAP) THEN                                            FRETRAC
C****     INITIATE FLAP RETRACTION.                                    FRETRAC
          INIFLAP    = .TRUE.                                          FRETRAC
          LINENUM  = LINENUM + 1                                       FRETRAC
          IF(IFLAP        .LE.0) IFLAP       = 1                       FRETRAC
          IF(DFLAPDT  .EQ.0.0) DFLAPDT  = 1.0                          FRETRAC
          IF(IFLAP.LE.MAXSIZF) THEN                                    FRETRAC
              FLAPNEW = FLPARY(IFLAP)                                  FRETRAC
          ELSE                                                         FRETRAC
              FLAPNEW = 0.0                                            FRETRAC
          ENDIF                                                        FRETRAC
          TIMEJ = TIME                                                 FRETRAC
          TIMEFLP = (FLAP - FLAPNEW)/DFLAPDT                           FRETRAC
          WRITE(LUOUT,1001) FLAPNEW,TIMEFLP                            FRETRAC
1001      FORMAT( ' FLAPS RETRACTED TO ',F4.1,' DEGREES IN ',          FRETRAC
     &              F4.1,' SECONDS.')                                  FRETRAC
      ENDIF                                                            FRETRAC
      IF(INIFLAP) THEN                                                 FRETRAC
          FLAP = FLAP - DFLAPDT*FLOAT(TIME - TIMEJ)                    FRETRAC
          IF(FLAP.LE.FLAPNEW) THEN                                     FRETRAC
              FLAP = FLAPNEW                                           FRETRAC
              IF(IFLAP.LE.MAXSIZF) THEN                                FRETRAC
                  IFLAP    = IFLAP + 1                                 FRETRAC
                  VKFLAP  = VFLPARY(IFLAP)                             FRETRAC
              ENDIF                                                    FRETRAC
              INIFLAP = .FALSE.                                        FRETRAC
              IF(FLAP.LE.0.0) FLAPFLG = .FALSE.                        FRETRAC
          ENDIF                                                        FRETRAC
      ENDIF                                                            FRETRAC
      TIMEJ = TIME                                                     FRETRAC
      RETURN                                                           FRETRAC
      END                                                              FRETRAC
```

Subroutine GRETRAC

```
      SUBROUTINE GRETRAC(DCDLGR,DTGEAR,HAGL)                                    GRETRAC
C**** THIS SUBROUTINE CONTROLS RETRACTION OF THE LANDING GEAR AND               GRETRAC
C**** RETURNS THE DELTA DRAG COEFFICIENT OF THE LANDING GEAR (DCDLGR)           GRETRAC
C**** BASED UPON THE DELTA TIME TO RETRACT THE GEAR (DTGEAR) AND ELAPSED        GRETRAC
C**** TIME FROM GEAR RETRACTION.                                                GRETRAC
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                         CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                                CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,   CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                                  CTRL
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,         FLAGS
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,           FLAGS
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,           FLAGS
     &          VFFLAG,WRITITR                                                  FLAGS
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,         FLAGS
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,         FLAGS
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,     FLAGS
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                          FLAGS
      REAL LGRARY(6)                                                            GEARDAT
      COMMON/GEARDAT/ IGEAR,MAXSIZG,LGRARY                                      GEARDAT
      LOGICAL INIGEAR                                                           GRETRAC
      IF(.NOT.INIGEAR) THEN                                                     GRETRAC
C****    INITIATE GEAR RETRACTION.                                              GRETRAC
         INIGEAR   = .TRUE.                                                     GRETRAC
         LINENUM   = LINENUM + 1                                                GRETRAC
         IF(IGEAR.LE.0) IGEAR = 1                                               GRETRAC
         TGEARUP = FLOAT(TIME) + DTGEAR                                         GRETRAC
         WRITE(LUOUT,1002) TIME,HAGL,TGEARUP                                    GRETRAC
1002     FORMAT( ' GEAR RETRACTION   (TIME = ',F6.2,'  HAGL = ',F8.1,           GRETRAC
     &           ' COMPLETE AT ',F6.2,' SECONDS)')                              GRETRAC
      ENDIF                                                                     GRETRAC
      IF(INIGEAR) THEN                                                          GRETRAC
         DGTIME  = TGEARUP - FLOAT(TIME)                                        GRETRAC
         IF(LGRARY(1).EQ.0.000) THEN                                            GRETRAC
C****       GEAR DRAG INCREMENT REDUCED LINEARLY WITH TIME IN DTGEAR            GRETRAC
C****       SECONDS.                                                            GRETRAC
            GRFACTR = DGTIME/DTGEAR                                             GRETRAC
            DCDLGR   = DCDLGR*GRFACTR                                           GRETRAC
            IF(DCDLGR.LE.0.0000) THEN                                           GRETRAC
C****          RESET VARIABLES AND FLAGS.                                       GRETRAC
               DCDLGR   = 0.0000                                                GRETRAC
               INIGEAR  = .FALSE.                                               GRETRAC
               LGRFLAG  = .FALSE.                                               GRETRAC
            ENDIF                                                               GRETRAC
```

```
            ELSE                                                            GRETRAC
C****           GEAR DRAG INCREMENT OBTAINED FROM USER INPUT ARRAY, LGRARY. GRETRAC
C****           EACH ELEMENT IN THE ARRAY IS ASSIGNED TO DCDLGR FOR ONE     GRETRAC
C****           SECOND UNTIL DCDLGR EQUALS ZERO OR IGEAR EXCEEDS MAXSIZG.   GRETRAC
                IGEAR = INT(DTGEAR - DGTIME) + 1                            GRETRAC
                IF(IGEAR.LE.MAXSIZG) THEN                                   GRETRAC
                    DCDLGR = LGRARY(IGEAR)                                  GRETRAC
                ELSE                                                        GRETRAC
C****               RESET VARIABLES AND FLAGS.                              GRETRAC
                    DCDLGR   = 0.0000                                       GRETRAC
                    INIGEAR  = .FALSE.                                      GRETRAC
                    LGRFLAG  = .FALSE.                                      GRETRAC
                ENDIF                                                       GRETRAC
            ENDIF                                                           GRETRAC
        ENDIF                                                               GRETRAC
        RETURN                                                              GRETRAC
        END                                                                 GRETRAC
```

```
      SUBROUTINE PITCH(MANUVR,ALPHA,DADTCMD,DTIME,DTDTGEX,LUOUT)          PITCH
C**** THIS SUBROUTINE MODULATES ANGLE OF ATTACK TO MATCH AN INPUT PITCH    PITCH
C**** ATTITUDE OR CLIMB SPEED IF LIMIT CONDITIONS ARE MET.  THE RATE AT    PITCH
C**** WHICH ANGLE OF ATTACK IS INCREASED IS BASED ON LOAD FACTOR (XLF)     PITCH
C**** AND AN INPUT FIRST DERIVATIVE OF ALPHA WITH RESPECT TO TIME          PITCH
C**** (DADTCMD).                                                           PITCH
C****                                                                      PITCH
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,  AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                  AIRCRFT
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ   AIRBORN
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG
     &                 ROC,RKAIR                                           FPINTEG
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG
     &                 ROC,RKAIR(40)                                       FPINTEG
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                     CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                     CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                     CONST
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,     FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,       FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,       FLAGS
     &         VFFLAG,WRITITR                                              FLAGS
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,     FLAGS
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,     FLAGS
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                      FLAGS
      CHARACTER MANUVR*7                                                   PITCH
      DOUBLE PRECISION DTIME                                               PITCH
      IF(DTDTGEX.LE.0.0) DTDTGEX = 1.0                                     PITCH
      IF    (DADTCMD.LT.0.0) THEN                                          PITCH
C****     REDUCE ANGLE OF ATTACK -- DADTCMD NEGATIVE.                      PITCH
          IF    (MANUVR.EQ.'CLIMB')     THEN                               PITCH
              IF(XLF.GE.0.85) THEN                                         PITCH
                  RFACTOR = 0.50*DTDTGEX                                   PITCH
              ELSE                                                         PITCH
                  RFACTOR = 0.25*DTDTGEX                                   PITCH
              ENDIF                                                        PITCH
          ELSEIF(MANUVR.EQ.'LANDING')  THEN                                PITCH
              RFACTOR = DTDTGEX                                            PITCH
          ELSEIF(MANUVR.EQ.'ROLL')     THEN                                PITCH
              RFACTOR = DTDTGEX*2.0                                        PITCH
          ELSEIF(MANUVR.EQ.'ROTATE')   THEN                                PITCH
              RFACTOR = 1.0                                                PITCH
          ELSE                                                             PITCH
              RFACTOR = 1.0                                                PITCH
          ENDIF                                                            PITCH
```

```
      ELSEIF(DADTCMD.GT.0.0) THEN
C****      INCREASE ANGLE OF ATTACK -- DADTCMD POSITIVE.
          IF    (MANUVR.EQ.'CLIMB  ')  THEN
              IF(THETAF.LE.THTMAX) THEN
                  IF    (XLF.GE.0.9)      THEN
                      RFACTOR = DTDTGEX
                  ELSEIF(XLF.LT.0.9 .AND. XLF.GE.0.8) THEN
                      RFACTOR = DTDTGEX*2.0
                  ELSE
                      RFACTOR = DTDTGEX*3.0
                  ENDIF
              ELSE
                  TERMFLG = .TRUE.
                  RFACTOR = 0.0
                  WRITE(LUOUT,1003) THETAF,THTMAX,XLF
              ENDIF
1003          FORMAT( ' THE PROGRAM HAS ATTEMPTED TO SIMULATE A CLIMB',
     &                ' OR PULLUP THAT IS BEYOND THE INPUT LIMITS OF'/,
     &                ' MAXIMUM PITCH ATTITUDE.'/,
     &                ' THETA, MAX THETA, LOAD FACTOR =', F10.2,
     &                ' DEGREES',F10.2,' DEGREES',F10.2,' G')
          ELSEIF(MANUVR.EQ.'LANDING')  THEN
              IF(XLF.GE.1.0) THEN
                  RFACTOR = DTDTGEX
              ELSE
                  RFACTOR = DTDTGEX*2.0
              ENDIF
          ELSEIF(MANUVR.EQ.'ROLL   ')     THEN
              RFACTOR = DTDTGEX
          ELSEIF(MANUVR.EQ.'ROTATE ')  THEN
              RFACTOR = 1.0
          ELSE
              RFACTOR = 1.0
          ENDIF
      ELSE
          TERMFLG = .TRUE.
          WRITE(LUOUT,1004)
1004      FORMAT(  ' NO PITCH MODULATION IS POSSIBLE.  DADTCMD IS SET',
     &             ' TO ZERO.')
      ENDIF
      ALPHA    = ALPHA + DADTCMD*FLOAT(DTIME)*RFACTOR
      GAMMA    = GAMMAR*RX
      THETAF   = ALPHA + GAMMA
      IF       (THETAF.GT.THTMAX .AND. (        LIFTOFF)) THEN
          THETAF    = THTMAX
          GAMMA     = THTMAX - ALPHA
          GAMMAR    = GAMMA/RX
      ELSEIF (THETAF.GT.THTMAX .AND. (.NOT.LIFTOFF)) THEN
          THETAF    = THTMAX
          ALPHA     = THTMAX
          GAMMAR    = 0.0
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SPDBRAK(ACTION,SPDBRK,SBKEND,DSBKDT)                    SPDBRAK
C**** THIS SUBROUTINE CONTROLS THE SPEED BRAKES BASED UPON THE CHARACTER SPDBRAK
C**** VARIABLE (ACTION) AND THE TIME RATE OF CHANGE OF SPEED BRAKE ANGLE  SPDBRAK
C**** (DSBKDT).  THE VALID VALUES FOR ACTION ARE: 'RESET  ',              SPDBRAK
C**** 'DEPLOY ', AND 'RETRACT'.                                           SPDBRAK
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                  CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                         CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                           CTRL
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,   FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,     FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,     FLAGS
     &         VFFLAG,WRITITR                                            FLAGS
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,  FLAGS
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,  FLAGS
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                   FLAGS
      CHARACTER ACTION*7                                                 SPDBRAK
      DOUBLE PRECISION TIMEJ                                             SPDBRAK
      LOGICAL INISBK                                                     SPDBRAK
      IF      (ACTION.EQ.'RESET  ')          THEN                        SPDBRAK
          INISBK     = .FALSE.                                           SPDBRAK
          SBKFLAG  = .FALSE.                                             SPDBRAK
          SPDBRK     = SBKEND                                            SPDBRAK
          RETURN                                                         SPDBRAK
      ELSEIF  (.NOT.INISBK)           THEN                               SPDBRAK
C****     INITIATE SPEED BRAKE DEPLOYMENT OR RETRACTION.                 SPDBRAK
          INISBK = .TRUE.                                                SPDBRAK
          LINENUM = LINENUM + 1                                          SPDBRAK
          IF(DSBKDT.EQ.0.0) DSBKDT = 90.0                                SPDBRAK
          TIMEJ       = TIME                                             SPDBRAK
          TIMESBK   = (SBKEND - SPDBRK)/DSBKDT                           SPDBRAK
          IF      (ACTION.EQ.'DEPLOY ')   THEN                           SPDBRAK
              WRITE(LUOUT,1005) SBKEND,TIMESBK                           SPDBRAK
          ELSEIF(ACTION.EQ.'RETRACT')  THEN                              SPDBRAK
              WRITE(LUOUT,1006) SBKEND,TIMESBK                           SPDBRAK
          ENDIF                                                          SPDBRAK
1005      FORMAT(  ' SPEED BRAKES DEPLOYED TO',F5.1,' DEGREES IN',F5.1,  SPDBRAK
     &              ' SECONDS.')                                         SPDBRAK
1006      FORMAT(  ' SPEED BRAKES RETRACTED TO',F5.1,' DEGREES IN',F5.1, SPDBRAK
     &              ' SECONDS.')                                         SPDBRAK
      ENDIF                                                              SPDBRAK
      IF(INISBK) THEN                                                    SPDBRAK
          IF(ACTION.EQ.'DEPLOY ') THEN                                   SPDBRAK
              SPDBRK = SPDBRK + DSBKDT*FLOAT(TIME - TIMEJ)               SPDBRAK
              IF(SPDBRK.GE.SBKEND) THEN                                  SPDBRAK
                  INISBK     = .FALSE.                                   SPDBRAK
                  SBKFLAG = .TRUE.                                       SPDBRAK
                  SPDBRK    = SBKEND                                     SPDBRAK
              ENDIF                                                      SPDBRAK
```

```
      ELSE
         SBKFLAG = .FALSE.
         SPDBRK = SPDBRK - DSBKDT*FLOAT(TIME - TIMEJ)
         IF(SBKEND.LE.0.0) SBKEND = 0.0
         IF(SPDBRK.LE.SBKEND) THEN
            IF(SPDBRK.LE.0.0) INISBK = .FALSE.
            SPDBRK = SBKEND
         ENDIF
      ENDIF
   ENDIF
   TIMEJ = TIME
   RETURN
   END
```

```
      SUBROUTINE SPOIL(ACTION,SPOILER,SPLREND,DSPLRDT)
C**** THIS SUBROUTINE CONTROLS THE SPOILERS BASED UPON THE CHARACTER
C**** VARIABLE (ACTION) AND THE TIME RATE OF CHANGE OF SPOILER ANGLE
C**** (DSPLRPDT). THE VALID VALUES FOR ACTION ARE: 'RESET  ',
C**** 'DEPLOY ', AND 'RETRACT'.
      DOUBLEPRECISION DTIME,DTIMEJ,TIME
      PARAMETER (LUIN=3,LUOUT=4)
      COMMON/CTRL/       DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,
     &                   NEQ,NPAGE,TIME,TIMEROL
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,
     &          VFFLAG,WRITITR
      COMMON/FLAGS/      AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,
     &                   FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,
     &                   REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,
     &                   TERMFLG,VECTFLG,VFFLAG,WRITITR
      CHARACTER ACTION*7
      DOUBLE PRECISION TIMEJ
      LOGICAL INISPLR
      IF      (ACTION.EQ.'RESET  ')          THEN
          INISPLR   = .FALSE.
          SPLFLAG   = .FALSE.
          SPOILER   = SPLREND
          RETURN
      ELSEIF  (.NOT.INISPLR)                 THEN
C****     INITIATE SPOILER DEPLOYMENT OR RETRACTION.
          INISPLR = .TRUE.
          LINENUM = LINENUM + 1
          IF(DSPLRDT.EQ.0.0) DSPLRDT = 90.0
          TIMEJ     = TIME
          TIMESPL   = (SPLREND - SPOILER)/DSPLRDT
          IF     (ACTION.EQ.'DEPLOY ')  THEN
              WRITE(LUOUT,1005) SPLREND,TIMESPL
          ELSEIF(ACTION.EQ.'RETRACT')  THEN
              WRITE(LUOUT,1006) SPLREND,TIMESPL
          ENDIF
1005      FORMAT(  ' SPOILERS DEPLOYED TO',F5.1,' DEGREES IN',F5.1,
     &             ' SECONDS.')
1006      FORMAT(  ' SPOILERS RETRACTED TO',F5.1,' DEGREES IN',F5.1,
     &             ' SECONDS.')
      ENDIF
      IF(INISPLR) THEN
          IF(ACTION.EQ.'DEPLOY ') THEN
              SPOILER = SPOILER + DSPLRDT*FLOAT(TIME - TIMEJ)
              IF(SPOILER.GE.SPLREND) THEN
                  INISPLR   = .FALSE.
                  SPLFLAG   = .TRUE.
                  SPOILER   = SPLREND
              ENDIF
```

```
      ELSE
        SPLFLAG = .FALSE.
        SPOILER = SPOILER - DSPLRDT*FLOAT(TIME - TIMEJ)
        IF(SPLREND.LE.0.0) SPLREND = 0.0
        IF(SPOILER.LE.SPLREND) THEN
          IF(SPOILER.LE.ZERO) INISPLR = .FALSE.
          SPOILER = SPLREND
        ENDIF
      ENDIF
    ENDIF
    TIMEJ = TIME
    RETURN
    END
```

```
      SUBROUTINE TVECTOR(VTANGLE,HVECT,VVECT,DVECTDT)                     TVECTOR
C**** THIS SUBROUTINE CONTROLS THE THRUST VECTORING BASED UPON THE          TVECTOR
C**** THRUST VECTORING TRIGGER ALTITUDE (HVECT), THE THRUST                 TVECTOR
C**** VECTORING TRIGGER AIRSPEED (VVECT), AND THE DELTA TIME FOR THRUST     TVECTOR
C**** VECTORING (DVECTDT).                                                  TVECTOR
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                     CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                            CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                              CTRL
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,     FLAGS
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,       FLAGS
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,       FLAGS
     &          VFFLAG,WRITITR                                              FLAGS
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,     FLAGS
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,     FLAGS
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                      FLAGS
      COMMON/VECTDAT/ IVECT,MAXSIZV,XNUARY(5),HVCTARY(5),VVCTARY(5)         VECTDAT
      DOUBLE PRECISION TIMEJ                                                TVECTOR
      LOGICAL INIVECT                                                       TVECTOR
      IF(.NOT.INIVECT) THEN                                                 TVECTOR
C****     INITIATE THRUST VECTOR CHANGE.                                    TVECTOR
          INIVECT = .TRUE.                                                  TVECTOR
          LINENUM = LINENUM + 1                                             TVECTOR
          IF(IVECT     .LE.0    ) IVECT      = 1                            TVECTOR
          IF(DVECTDT .EQ.0.0 ) DVECTDT = 1.0                                TVECTOR
          IF(IVECT.LE.MAXSIZV) THEN                                         TVECTOR
              XNUNEW = XNUARY(IVECT)                                        TVECTOR
          ELSE                                                              TVECTOR
              XNUNEW = 0.0                                                  TVECTOR
          ENDIF                                                             TVECTOR
          TIMEJ     = TIME                                                  TVECTOR
          TIMEVCT = (XNUNEW - VTANGLE)/DVECTDT                              TVECTOR
          IF(TIMEVCT.LT.0.0) THEN                                           TVECTOR
C****         ADJUST SIGN OF TIMEVCT AND DVECTDT SO THAT TIMEVCT IS ALWAYS  TVECTOR
C****         POSITIVE AND DVECTDT IS POSITIVE IF THE VECTORED THRUST       TVECTOR
C****         ANGLE IS INCREASING AND DVECTDT IS NEGATIVE IF THE VECTOR     TVECTOR
C****         THRUST ANGLE IS DECREASING.                                   TVECTOR
              TIMEVCT  = -TIMEVCT                                           TVECTOR
              DVECTDT  = -DVECTDT                                           TVECTOR
          ENDIF                                                             TVECTOR
          WRITE(LUOUT,1007) VTANGLE,XNUNEW,TIMEVCT                          TVECTOR
1007      FORMAT( ' VECTORED THRUST ANGLE CHANGED FROM ',F4.1,' TO ',F4.1,  TVECTOR
     &                ' DEGREES IN',F4.1,' SECONDS.')                       TVECTOR
      ENDIF                                                                 TVECTOR
```

```
IF(INIVECT) THEN                                              TVECTOR
      VTANGLE = VTANGLE + DVECTDT*FLOAT(TIME - TIMEJ)         TVECTOR
      IF(VTANGLE.LE.XNUNEW) THEN                              TVECTOR
            VTANGLE = XNUNEW                                  TVECTOR
            IF(IVECT.LE.MAXSIZV) THEN                         TVECTOR
                  IVECT    = IVECT + 1                        TVECTOR
                  HVECT    = HVCTARY(IVECT)                   TVECTOR
                  VVECT    = VVCTARY(IVECT)                   TVECTOR
            ENDIF                                             TVECTOR
            INIVECT = .FALSE.                                 TVECTOR
            IF(VTANGLE.EQ.0.0) VECTFLG = .FALSE.              TVECTOR
      ENDIF                                                   TVECTOR
ENDIF                                                         TVECTOR
TIMEJ = TIME                                                  TVECTOR
RETURN                                                        TVECTOR
END                                                           TVECTOR
```

```
      SUBROUTINE TAKOFF(ALPHA,GDIST)                                        TAKOFF
C**** THIS SUBROUTINE CONTROLS THE EXECUTION OF THE TAKEOFF MANEUVER        TAKOFF
C**** FROM BRAKE RELEASE THROUGH CLIMBOUT.  SUBROUTINE TAKEOFF CALLS INTX    TAKOFF
C**** TO PERFORM THE NUMERICAL INTEGRATION OF THE RESULTANTS OF THE         TAKOFF
C**** EQUATIONS OF MOTION AND CALLS FORCEX TO OBTAIN THE FORCE              TAKOFF
C**** COEFFICIENTS FOR THE EQUATIONS OF MOTION.                             TAKOFF
C****                                                                       TAKOFF
C**** THE CONTINUED TAKEOFF IS DONE BY ACCELERATING TO SPEED VKFAIL,        TAKOFF
C**** LINEARLY DECREASING WITH TIME THE NUMBER OF ENGINES FROM A VALUE      TAKOFF
C**** OF XENG TO A VALUE OF XENGOUT, CONTINUING THE GROUND ROLL TO          TAKOFF
C**** VKROTAT, ROTATE (IF VKFAIL IS LESS THAN VKROTAT), LIFTOFF AND         TAKOFF
C**** CLIMBOUT TO HMAX.  VKFAIL MUST BE LESS THAN TAKEOFF SPEED FOR AN      TAKOFF
C**** ENGINE FAILURE TO BE INITIATED.                                       TAKOFF
C****                                                                       TAKOFF
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                     CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                            CTRL
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                 NEQ,NPAGE,TIME,TIMEROL                               CTRL
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,   AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                   AIRCRFT
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,      AERO
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                           AERO
      COMMON/ENGINE/   AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,       ENGINE
     &                 REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,         ENGINE
     &                 XENGOUT,XIDLE,XMIL,ZFN                               ENGINE
      COMMON/VECTOR/   HVECT,VVECT                                          VECTOR
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ    AIRBORN
      COMMON/AIRSPED/  VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,    AIRSPED
     &                 VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                   AIRSPED
      COMMON/RUNWAY/   ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,       RUNWAY
     &                 HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,  RUNWAY
     &                 TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                  RUNWAY
      DOUBLEPRECISION  VTAS,DIST,ACCEL,VTASJ,RKGRND                         INTEG
      COMMON/INTEG/    VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                     INTEG
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,     FPINTEG
     &                 ROC,RKAIR                                            FPINTEG
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,     FPINTEG
     &                 ROC,RKAIR(40)                                        FPINTEG
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF ATMOS
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                      CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                      CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                      CONST
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,      FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,        FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,        FLAGS
     &         VFFLAG,WRITITR                                               FLAGS
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,      FLAGS
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,      FLAGS
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,  FLAGS
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                       FLAGS
```

```
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,      CHARV
     &          THRCRV*3,TKOTYPE*7                                         CHARV
      COMMON/CHARV/    ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,    CHARV
     &                 TKOTYPE                                             CHARV
      COMMON/FLAPDAT/  IFLAP,MAXSIZF,FLPARY(5),VFLPARY(5)                  FLAPDAT
      REAL LGRARY(6)                                                       GEARDAT
      COMMON/GEARDAT/  IGEAR,MAXSIZG,LGRARY                                GEARDAT
      COMMON/VECTDAT/  IVECT,MAXSIZV,XNUARY(5),HVCTARY(5),VVCTARY(5)       VECTDAT
      LOGICAL INIVCLM,INISEG1,INISEG2,VCOFLAG                              TAKOFF
      REAL ATMARY(8),S(48),T(24),INTERP                                    TAKOFF
      EQUIVALENCE (TEMPR,ATMARY(1)),(FPVTAS,S(1)),(VTAS,T(1))              TAKOFF
      EXTERNAL DERIVGR,DERIVAT,INTERP                                      TAKOFF
      DATA DISTMAX,HCLIMB, HCLMOUT,  HGEAR,HMAX, ROLLMAX,  THTCLM, THTFLY, TAKOFF
     &    /  60760.,   100.,     200.,    205., 1000.,    120.0,     10.0,      8.0,/,  TAKOFF
     &       THTROT, THTTOL, TIMEMAX,VCLMOUT,  VKEND, VKFAIL,   VKFLAP VKROTAT      TAKOFF
     &    /    10.0,      0.1,      300.,         0.,    250.,      0.,         0.,         0./,  TAKOFF
     &        VKSTART                                                      TAKOFF
     &    /        0./                                                     TAKOFF
      NAMELIST/TKO/HGEAR,HMAX,THTCLM,THTROT,VKFAIL,VKROTAT                 TAKOFF
      NAMELIST/TKO2/DISTMAX,HCLIMB,HCLMOUT,ROLLMAX,THTFLY,THTTOL,          TAKOFF
     &              TIMEMAX,TKOTYPE,VCLMOUT,VKEND,VKFLAP,VKSTART           TAKOFF
      NAMELIST/TKOARY/FLPARY,VFLPARY,HVCTARY,VVCTARY,XNUARY                TAKOFF
C****                                                                      TAKOFF
C**** INPUT DATA LOADED INTO TAKOFF THRU NAMELISTS /TKO/ AND /TKOARY/.     TAKOFF
      READ(LUIN,TKO)                                                       TAKOFF
      READ(LUIN,TKO2)                                                      TAKOFF
      READ(LUIN,TKOARY)                                                    TAKOFF
C****                                                                      TAKOFF
C**** TERMINATE PROGRAM IF ROTATION SPEED NOT INITIALIZED.                 TAKOFF
      IF(VKROTAT.EQ.ZERO) CALL HALT( LUIN,LUMSG,LUOUT,                     TAKOFF
      &                         ' VKROTAT NOT INPUT BY USER.')             TAKOFF
C****                                                                      TAKOFF
C**** INITIALIZE INTERNAL SUBROUTINE LOGIC CONTROL VARIABLES.              TAKOFF
      INIVCLM = INISEG1 = INISEG2 = LIFTOFF = ROTATE = VCOFLAG = .FALSE.   TAKOFF
C****                                                                      TAKOFF
C**** INITIALIZE LOGIC FLAG BASED ON ENGINE FAILURE SPEED.                 TAKOFF
      IF(VKFAIL.EQ.ZERO) THEN                                              TAKOFF
          FAILFLG = .FALSE.                                                TAKOFF
      ELSE                                                                 TAKOFF
          FAILFLG = .TRUE.                                                 TAKOFF
      ENDIF                                                                TAKOFF
C****                                                                      TAKOFF
C**** INITIALIZE CURVE FILES.                                              TAKOFF
      CALL INICURV                                                         TAKOFF
C****                                                                      TAKOFF
C**** INITIALIZE INTERNAL SUBROUTINE PARAMETERS.                           TAKOFF
      TEMPF   = TSLF + DTEMPF - 0.003566*HRUNWAY                           TAKOFF
      THTMAX = THTROT                                                      TAKOFF
```

```
C****                                                                   TAKOFF
C**** OBTAIN ATMOSPHERIC VARIABLES.                                     TAKOFF
      CALL ATMOSPH(HRUNWAY,ATMARY)                                      TAKOFF
C****                                                                   TAKOFF
C**** SUM VKSTART AND VKWIND TO CALCULATE INITIAL AIRSPEED FOR ROLLING   TAKOFF
C**** MINIMUM INTERVAL TAKEOFFS.                                        TAKOFF
      VKTAS= VKWIND + VKSTART                                           TAKOFF
      VTAS  = VKTAS*FPSKTS                                              TAKOFF
C****                                                                   TAKOFF
C**** CALIBRATED AIRSPEED EQUATION                                      TAKOFF
C**** ASLSQR5 = 661.48*SQRT(5.0); TWOOVR7 = 2.0/7.0                     TAKOFF
C**** VKCAS    = ASL*SQRT(5.0*(DELTA*                                   TAKOFF
C****&          ((1.0 + 0.2*AMACH**2)**3.5 - 1.0) + 1.0)**(2.0/7.0) - 1.0))  TAKOFF
      CALL SPEED(0.0,VTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,VKTGS,VTGS)  TAKOFF
C**** OBTAIN INITIAL CONDITIONS OF THRUST AND FORCE COEFFICIENTS FOR    TAKOFF
C**** GROUND ROLL.                                                      TAKOFF
      CALL FORCEX(ALPHA,CD,CL)                                          TAKOFF
C****                                                                   TAKOFF
C**** OBTAIN STATIC THRUST TO WEIGHT RATIO.                             TAKOFF
      TOVERW = THRUST*(FLOAT(NENG)/XENG)/GWT                            TAKOFF
C****                                                                   TAKOFF
C**** INITIALIZE THRUST ANGLE SCHEDULES FOR AIRBORNE PORTION OF         TAKOFF
C**** TAKEOFF.                                                          TAKOFF
      HVECT    = HVCTARY(1)                                             TAKOFF
      VVECT    = VVCTARY(1)                                             TAKOFF
      VTANGLE = XNUARY(1)                                               TAKOFF
C****                                                                   TAKOFF
C**** REINITIALIZE THE VVCTARY ARRAY ELEMENTS FOR ANY CORRESPONDING     TAKOFF
C**** ELEMENTS IN THE HVCTARY ARRAY.  THIS GIVES ALTITUDE BREAKPOINTS   TAKOFF
C**** PRECEDENCE OVER AIRSPEED BREAKPOINTS FOR THRUST VECTORING         TAKOFF
C**** MODULATION.                                                       TAKOFF
      DO 10 I = 2,MAXSIZV                                               TAKOFF
         IF(HVCTARY(I).NE.ZERO) VVCTARY(I) = ZERO                       TAKOFF
   10 CONTINUE                                                          TAKOFF
C****                                                                   TAKOFF
C**** INITIALIZE FLAPS FOR GROUND ROLL.                                 TAKOFF
      IF    (VFLPARY(1)    .NE.ZERO) THEN                               TAKOFF
         VKFLAP = VFLPARY(1)                                            TAKOFF
      ELSEIF(VKFLAP .EQ.ZERO) THEN                                      TAKOFF
         VKFLAP = VKFLPMX                                               TAKOFF
      ENDIF                                                             TAKOFF
```

```
C****
C**** ECHO BACK PROGRAM INPUTS TO OUTPUT FILE.
      IF(VKWIND.GE.ZERO) THEN
          WRITE(LUOUT,1001) HRUNWAY,GAMMARW,TEMPF,        VKWIND,'HEADWIND'
      ELSE
          WRITE(LUOUT,1001) HRUNWAY,GAMMARW,TEMPF,ABS(VKWIND),'TAILWIND'
      ENDIF
      WRITE(LUOUT,1002) GWT,THTROT,SWING,THTCLM,GWT*TOVERW/DELTA,THTFLY,
     &                  WNGLOD,VCLMOUT,TOVERW,HGEAR,XENG,DADTCMD,DTFAIL
      WRITE(LUOUT,1003)
     &    (XNUARY  (I),I = 1,MAXSIZV),
     &    (VVCTARY (I),I = 1,MAXSIZV),(HVCTARY(I),I = 1,MAXSIZV),
     &    (FLPARY  (I),I = 1,MAXSIZF),(VFLPARY(I),I = 1,MAXSIZF),
     &    HRUNWAY
      WRITE(LUOUT,1004)
1001 FORMAT( '** INPUTS TO TAKE OFF **'/, ALTITUDE =',F7.1,
     &          '      ALTITUDE =',F7.1,19X,' RUNWAY SLOPE = ',F4.1,' DEGREES
     &          ' DEGREES'/,
     &          ' TEMPERATURE = ',F7.1,' DEGREES F'/,
     &          '                  ',F7.1,' KNOT ',A8)
1002 FORMAT( /,' AIRCRAFT PARAMETERS',20X,
     &           'FLIGHTPATH CONTROL PARAMETERS'/,
     &          /,' GROSS RAMP WEIGHT        =',F9.0,2X,
     &           ' ROTATION PITCH ANGLE         =',F6.1,
     &          /,' WING AREA =               =',F9.0,2X,
     &           ' SEGMENT I PITCH ANGLE        =',F6.1,
     &          /,' STATIC SEA LEVEL THRUST =',F9.0,2X,
     &           ' SEGMENT II PITCH ANGLE       =',F6.1,
     &          /,' WING LOADING             =',F9.1,2X,
     &           ' CLIMB OUT AIRSPEED           =',F6.1,
     &          /,' THRUST/WEIGHT            =',F9.3,2X,
     &           ' GEAR RETRACTION ALTITUDE   =',F6.1,
     &          /,' NUMBER OF ENGINES        =',F9.1,2X,
     &           ' COMMANDED ALPHA RATE         =',F6.1,
     &          /,'                          ',11X,
     &           ' ELAPSED ENGINE FAILURE TIME =',F6.1)
1003 FORMAT(/,' FLAP AND VECTORED THRUST SCHEDULES'/,
     &          9X,'FLAP RETRACTION SCHEDULE'/,
     &          12X,'FLAP DEFLECTION',5F9.1/,
     &          12X,'SPEED          ',5F9.1/,
     &          9X,'VECTORED THRUST ANGLE'/,12X,'ANGLE  ', 5F9.1/,
     &          12X,'SPEED  ',5F9.1/,12X,'ALTITUDE',5F9.0/,
     &          9X,'ALL SPEEDS ARE CALIBRATED AIR SPEEDS AND ALL ',
     &          'ALTITUDES ARE ABSOLUTE ALTITUDES.'/,
     &          '1 TAKEOFF (ELEVATION =',F6.0,' FEET)'//)
1004 FORMAT(' TIME  DIST.  WEIGHT   ALT.  VCAS  VTAS  VTGS',
     &        ' ACCEL    CL    CD  THETA  ALPHA  GAMMA  DTHET ',
     &        '  R/C  LOAD  THRUST  XENG'/,
     &        ' (SEC)  (FEET)   (LBS)  (FEET)  (KTS)  (KTS)  (KTS)',
     &        ' (FPS2)          (DEG)  (DEG)  (DEG)   /DT ',
     &        ' (FPM)  FACT   (LBS)  OR MU'/)
```

```
C****
C**** GROUND ROLL INTEGRATION VARIABLES.
C**** NEQ   = NUMBER OF EQUATIONS
C**** DTIME = TIME INTERVAL, STEP SIZE (SECONDS)
C**** VTAS  = VELOCITY (FEET/SECOND) WITH RESPECT TO THE AIR
C**** DIST  = DISTANCE (FEET)
C**** ACCEL = ACCELERATION (FEET/SECOND**2)
      GDIST = ZERO
      CALL INTX(NEQ,TIME,DTIME,T,DERIVGR,ALPHA)
      IF(DTIME.GE.0.10D0) THEN
          WRITE(LUOUT,1005) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,THETAF,ACCEL,
     &                      CL,CD,ZERO,ALPHA,ZERO,ZERO,ZERO,XLF,THRUST,
     &                      XENG
      ELSE
          WRITE(LUOUT,1006) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,THETAF,ACCEL,
     &                      CL,CD,ZERO,ALPHA,ZERO,ZERO,ZERO,XLF,THRUST,
     &                      XENG
      ENDIF
1005  FORMAT(F6.1,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,F6.2,
     &       F9.0,F7.3)
1006  FORMAT(F6.2,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,F6.2,
     &       F9.0,F7.3)
C****
C**** GROUND ROLL INTEGRATION LOOP
   19 DO 20 NCOUNT=1,10
          ALPHAJ = ALPHA
          THETAJ = THETAF
C****
C****     LIFTOFF CRITERION; IF LIFT IS LESS THAN WEIGHT
          IF    (QS*CY.LT.GWT*COSD(GAMMARW)) THEN
C****          LIFTOFF CRITERION NOT MET
C****
C****          MAKE INTEGRATION STEP.
               CALL INTZ(NEQ,TIME,DTIME,T,DERIVGR,ALPHA)
               CALL INTG( DIST,RKGRND(14),DTIME,0.0,VWIND,WFUEL,
                          DDIST,GDIST,GWT)
```

```
C****                                                              TAKOFF
C****     CALCULATE DYNAMIC PRESSURE, MACH NUMBER AND VELOCITIES.  TAKOFF
          CALL SPEED( 0.0,VTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,   TAKOFF
     &                VKTGS,VTGS)                                  TAKOFF
C****                                                              TAKOFF
C****     CHECK SPEED FOR VKFAIL.                                  TAKOFF
          IF(VKCAS.GE.VKFAIL .AND. (.NOT.VFFLAG)) THEN             TAKOFF
C****        ENGINE FAILURE SPEED ATTAINED.                        TAKOFF
             SPOOL   = .TRUE.                                      TAKOFF
             VFFLAG  = .TRUE.                                      TAKOFF
             TIMEFLD= FLOAT(TIME - DTIME)                          TAKOFF
             IF(.NOT.RTOFLAG) THEN                                 TAKOFF
                XENGTRN = 1.0                                      TAKOFF
                IF     (FAILST.EQ.'IDLE') THEN                     TAKOFF
                   XENGEND = XIDLE                                 TAKOFF
                ELSEIF (FAILST.EQ.' MIL') THEN                     TAKOFF
                   XENGEND = XMIL                                  TAKOFF
                ELSEIF (FAILST.EQ.' OFF') THEN                     TAKOFF
                   XENGEND = ZERO                                  TAKOFF
                ELSE                                               TAKOFF
                   XENGEND = 1.0 -XENGFLD                          TAKOFF
                ENDIF                                              TAKOFF
                XENGF = 1.0                                        TAKOFF
             ENDIF                                                 TAKOFF
             IF(FAILFLG) THEN                                      TAKOFF
C****           ENGINE FAILURE OCCURS.                             TAKOFF
C****           PARAMETER ENGGRP SWITCHED TO FAILGRP FOR           TAKOFF
C****           CONTINUED OR REFUSED PORTION OF TAKEOFF.           TAKOFF
                LINENUM  = LINENUM + 1                             TAKOFF
                ENGGRP   = FAILGRP                                 TAKOFF
                CALL INICURV                                       TAKOFF
                WRITE(LUOUT,1007) TIME,GDIST,VKCAS,VKTAS           TAKOFF
1007            FORMAT( ' ENGINE FAILURE    (TIME = ',F6.2,        TAKOFF
     &                  ' DIST = ',F8.1,' CAS = ',F7.1,            TAKOFF
     &                  ' TAS = ',F7.1,' KNOTS)')                  TAKOFF
             ENDIF                                                 TAKOFF
          ENDIF                                                    TAKOFF
C****                                                              TAKOFF
C****     ENGINE FAILURE CONTROL                                   TAKOFF
          IF(ENGGRP.EQ.FAILGRP) THEN                               TAKOFF
             IF      (RTOFLAG)THEN                                 TAKOFF
C****           REFUSED TAKEOFF                                    TAKOFF
C****           THIS FLAG RETURNS EXECUTION TO THE CALLING PROGRAM TAKOFF
C****           AT START OF THE ENGINE FAILURE. PROGRAM            TAKOFF
C****           EXECUTION WILL PROCEED WITH SUBROUTINE ROLL.       TAKOFF
                ICOUNT = NCOUNT + 1                                TAKOFF
                RETURN                                             TAKOFF
```

```
                ELSEIF (FAILMOD.EQ.'SEIZE') THEN                          TAKOFF
                   SPOOL = .TRUE.                                         TAKOFF
                   XENG = XENGOUT                                         TAKOFF
                ELSE                                                      TAKOFF
                   IF      (DTFAIL.EQ.ZERO .AND. (.NOT. SPOOL)) THEN      TAKOFF
                      CALL SPOOLDNF(TIME,XENGEND,XENGTRN,SPOOL,XENGF,     TAKOFF
     &                              LUMSG)                                TAKOFF
                      XENG = XENGOUT + XENGF                              TAKOFF
                   ELSEIF(DTFAIL.NE.ZERO .AND. (.NOT. SPOOL)) THEN        TAKOFF
                      XENG = XENG - XENGFLD*FLOAT(DTIME)/DTFAIL           TAKOFF
                      IF(XENG.LE.XENGOUT) THEN                            TAKOFF
                         SPOOL  = .TRUE.                                  TAKOFF
                      ENDIF                                               TAKOFF
                   ELSE                                                   TAKOFF
                      XENG  = XENGOUT                                     TAKOFF
                   ENDIF                                                  TAKOFF
                ENDIF                                                     TAKOFF
             ENDIF                                                        TAKOFF
C****        CHECK SPEED FOR VKROTAT.                                     TAKOFF
             IF(VKCAS.GE.VKROTAT) THEN                                    TAKOFF
                IF(.NOT.ROTATE) THEN                                      TAKOFF
C****                BEGIN ROTATION                                       TAKOFF
                     ROTATE   = .TRUE.                                    TAKOFF
                     AOA0FLG = .FALSE.                                    TAKOFF
                     LINENUM  = LINENUM + 1                               TAKOFF
                     WRITE(LUOUT,1008) TIME,GDIST,VKCAS,VKTGS             TAKOFF
1008                 FORMAT( ' ROTATION       (TIME = ',F6.2,             TAKOFF
     &                       ' DIST = ',F8.1,' CAS = ',F7.1,              TAKOFF
     &                       '  VG = ',F7.1,' KNOTS)')                    TAKOFF
                ENDIF                                                     TAKOFF
C****                                                                     TAKOFF
C****           INCREASE ALPHA TO TAIL SCRAPE ANGLE AT RATE DADTCMD.      TAKOFF
                CALL PITCH('ROTATE ',ALPHA,DADTCMD,DTIME,DTDTGEX,LUOUT)   TAKOFF
             ENDIF                                                        TAKOFF
C****                                                                     TAKOFF
C****        CHECK TIME FOR LIMIT OF GROUND ROLL TIME.                    TAKOFF
             IF(FLOAT(TIME).GT.ROLLMAX) THEN                              TAKOFF
                WRITE(LUOUT,1009)                                         TAKOFF
1009            FORMAT(/,' TIME LIMIT FOR GROUND RUN EXCEEDED.')          TAKOFF
                RETURN                                                    TAKOFF
             ENDIF                                                        TAKOFF
```

```
                IF(NCOUNT.EQ.10) THEN                                          TAKOFF
C****                CALCULATE ADDITIONAL OUTPUT PARAMETERS, WRITE OUTPUT AND   TAKOFF
C****                INCREMENT LINE NUMBER COUNTER LINENUM.                     TAKOFF
C****                                                                           TAKOFF
C****                CALCULATE LOCAL(BASED ON THE CURRENT AND PREVIOUS TIME     TAKOFF
C****                POINTS) RATES OF CHANGE IN ANGLE OF ATTACK AND PITCH       TAKOFF
C****                ATTITUDE. THESE ARE NOT THE COMMANDED RATES OF CHANGE      TAKOFF
C****                IN ANGLE OF ATTACK AND PITCH ATTITUDE.                     TAKOFF
                     DADT    = (ALPHA  - ALPHAJ)/FLOAT(DTIME)                   TAKOFF
                     DTHTDT = (THETAF - THETAJ)/FLOAT(DTIME)                    TAKOFF
                     CALL FORCEX(ALPHA,CD,CL)                                   TAKOFF
                     IF(DTIME.GE.0.10D0)THEN                                    TAKOFF
                          WRITE(LUOUT,1005) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,    TAKOFF
     &                              VKTGS,ACCEL,CL,CD,THETAF,ALPHA,ZERO,        TAKOFF
     &                              DTHTDT,ZERO,XLF,THRUST,XENG                 TAKOFF
                     ELSE                                                       TAKOFF
                          WRITE(LUOUT,1006) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,    TAKOFF
     &                              VKTGS,ACCEL,CL,CD,THETAF,ALPHA,ZERO,        TAKOFF
     &                              DTHTDT,ZERO,XLF,THRUST,XENG                 TAKOFF
                     ENDIF                                                      TAKOFF
                     LINENUM = LINENUM + 1                                      TAKOFF
                     IF(LINENUM.GE.NPAGE) THEN                                  TAKOFF
C****                     RESET LINE NUMBER COUNTER LINENUM AND WRITE HEADER    TAKOFF
C****                     FOR NEW PAGE.                                         TAKOFF
                          LINENUM = 5                                           TAKOFF
                          WRITE(LUOUT,1010)                                     TAKOFF
                          WRITE(LUOUT,1004)                                     TAKOFF
 1010                     FORMAT('1 TAKEOFF CONTINUED',/)                       TAKOFF
                     ENDIF                                                      TAKOFF
C****                RESTART GROUND ROLL INTEGRATION LOOP.                      TAKOFF
                     GO TO 19                                                   TAKOFF
                ENDIF                                                           TAKOFF
            ELSEIF(.NOT.LIFTOFF) THEN                                           TAKOFF
C****           LIFTOFF CRITERION MET. INITIALIZE ICOUNT FOR AIRBORNE           TAKOFF
C****           INTEGRATION LOOP. THE VARIABLE ICOUNT, ALLOWS SUBROUTINE        TAKOFF
C****           TAKOFF TO WRITE OUTPUT AT DTIME*10.0 SECOND INTERVALS AS        TAKOFF
C****           PROGRAM EXECUTION PROCEEDS FROM THE GROUND ROLL INTEGRATION     TAKOFF
C****           LOOP TO THE AIRBORNE INTEGRATION LOOP OR TO THE GROUND ROLL     TAKOFF
C****           INTEGRATION LOOP IN SUBROUTINE ROLL FOR REFUSED TAKEOFFS.       TAKOFF
                ICOUNT   = NCOUNT                                               TAKOFF
                LIFTOFF  = .TRUE.                                               TAKOFF
            ENDIF                                                               TAKOFF
   20 CONTINUE                                                                  TAKOFF
```

```
C****
C**** OUTPUT VARIABLES AT LIFTOFF.
      LINENUM  = LINENUM + 1
      WRITE(LUOUT,1011) TIME,GDIST,VKCAS,VKTGS
1011 FORMAT(' LIFTOFF          (TIME = ',F6.2,'  DIST = ',F8.1,
   &              ' CAS = ',F7.1,'  VG = ',F7.1,' KNOTS)')
C****
C**** AIRBORNE PORTION OF TAKEOFF
C****
C**** INITIALIZE THRUST ANGLE SCHEDULES FOR AIRBORNE PORTION OF TAKEOFF.
      HVECT = HVCTARY(1)
      VVECT = VVCTARY(1)
C****
C**** INITIALIZE AIRBORNE INTEGRATION VARIABLES.
C**** FPVTAS    = VELOCITY ALONG FLIGHTPATH (FEET/SECOND)
C**** GAMMAR  = FLIGHTPATH ANGLE (RADIANS)
C**** FPDIST    = DISTANCE ALONG FLIGHTPATH (FEET)
C**** PRESALT  = ALTITUDE (FEET)
C**** FPACCEL  = ACCELERATION ALONG FLIGHTPATH (FEET/SECOND**2)
C**** DGDTR     = TIME RATE OF CHANGE OF FLIGHTPATH ANGLE (RADIANS/SECOND)
C**** VHAS       = HORIZONTAL AIRSPEED (FEET/SECOND)
C**** ROC         = RATE OF CLIMB (FEET/SECOND)
      NEQ         = 4
      FPDIST     = DIST
      FPVTAS     = VTAS
      GAMMAR   = ZERO
      RKAIR(27) = RKGRND(14)
      CALL INTX(NEQ,TIME,DTIME,S,DERIVAT,ALPHA)
      CALL INTG(  DIST,RKGRND(14),DTIME,GAMMAR,VWIND,WFUEL,
                       DDIST,GDIST,GWT)
C****
C**** AIRBORNE INTEGRATION LOOP
      IF(ICOUNT.EQ.11) ICOUNT = 1
  29 DO 30 NCOUNT=ICOUNT,10
           ALPHAJ = ALPHA
           THETAJ = THETAF
           HAGL    = PRESALT - HRUNWAY
C****
C****      MAKE INTEGRATION STEP.
           CALL INTZ(NEQ,TIME,DTIME,S,DERIVAT,ALPHA)
           IF(ERRFLAG) THEN
                CALL ERROR(ROCFPM)
                RETURN
           ENDIF
           CALL INTG( DIST,RKAIR(27),DTIME,GAMMAR,VWIND,WFUEL,
                          DDIST,GDIST,GWT)
C****
C****      OBTAIN ATMOSPHERIC VARIABLES.
           CALL ATMOSPH(PRESALT,ATMARY)
```

```
C****
C****      CALCULATE DYNAMIC PRESSURE, MACH NUMBER AND VELOCITIES.
           CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,
                         VKTGS,VTGS)
C****
C****      CHECK FOR CLEARANCE HEIGHT HCLEAR.
           IF     ((.NOT.CLRHGT) .AND. HAGL.LT.HCLEAR) THEN
C****          STORE VALUES FOR HCLEAR FEET INTERPOLATION
              FPVTASJ  = FPVTAS
              GDISTJ   = GDIST
              HAGLJ    = HAGL
           ELSEIF((.NOT.CLRHGT) .AND. HAGL.GE.HCLEAR) THEN
C****          FIND VALUES AT HCLEAR FEET AGL.
               CLRHGT  = .TRUE.
               LINENUM = LINENUM + 1
               GD50    = INTERP(GDIST   ,GDISTJ  ,HAGL,HAGLJ,HCLEAR)
               VTAS50  = INTERP(FPVTAS,FPVTASJ ,HAGL,HAGLJ,HCLEAR)
               CALL SPEED(GAMMAR,VTAS50,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,
     &                    VKTGS,VTGS)
               WRITE(LUOUT,1012) HCLEAR,TIME,GD50,VKCAS,VKTAS
1012           FORMAT(' ALTITUDE:',F5.0,'FEET (TIME = ',F6.2,
     &                ' DIST = ',F8.1,' CAS = ',F7.1,' TAS = ',F7.1,
     &                ' KNOTS)')
           ENDIF
C****
C****      USE DEFAULT CLIMB OUT SPEED IF NO USER INPUT.  SET VCOFLAG TO
C****      .TRUE. TO PREVENT RE-INITIALIZING VCLMOUT.
           IF(.NOT.VCOFLAG) THEN
               IF(VCLMOUT.EQ.ZERO) VCLMOUT = VKCAS
               VCOFLAG = .TRUE.
           ENDIF
           IF     (HAGL.LT.HCLIMB)          THEN
C****          CONSTANT THETA PORTION OF TAKEOFF.
C****          MODULATE ALPHA FOR CONSTANT PITCH ANGLE CLIMB OR TO COMPLETE
C****          ROTATION.
               IF(.NOT.INISEG1) THEN
C****              INITIATE CONSTANT PITCH ANGLE CLIMB AT SEGMENT ONE CLIMB
C****              GRADIENT.  SET MAXIMUM PITCH ANGLE TO THTCLM FOR CONSTANT
C****              THETA PORTION OF TAKEOFF.
                   INISEG1  = .TRUE.
                   THTMAX   = THTCLM
               ENDIF
C****          CONSTANT THETA PORTION OF TAKEOFF
C****          MODULATE ALPHA FOR CONSTANT THETA CLIMB OR TO COMPLETE
C****          ROTATION.
               CALL PITCH('ROTATE ',ALPHA,DADTCMD,DTIME,DTDTGEX,LUOUT)
```

```
      ELSEIF(HAGL.GE.HCLIMB .AND. HAGL.LT.HCLMOUT) THEN
C****     CONSTANT THETA TO CONSTANT CLIMBOUT SPEED TRANSITION
C****     MODULATE ALPHA FOR CONSTANT SPEED CLIMB OUT WITHIN
C****     ALTITUDES HCLIMB AND HCLMOUT.
          IF(.NOT.INIVCLM) THEN
C****         INITIATE CONSTANT CALIBRATED AIRSPEED CLIMB.
              INIVCLM = .TRUE.
              THTMAX = 89.9
              LINENUM = LINENUM + 1
              WRITE(LUOUT,1012) HCLIMB,TIME,GDIST,VKCAS,VKTAS
          ENDIF
          IF(ROCFPM.GE.ZERO) THEN
              IF(VKCAS.LT.VCLMOUT) THEN
C****             REDUCE ALPHA
                  CALL PITCH('CLIMB ',ALPHA,-DADTCMD,DTIME,DTDTGEX,
     &                       LUOUT)
              ELSE
C****             INCREASE ALPHA
                  CALL PITCH('CLIMB ',ALPHA, DADTCMD,DTIME,DTDTGEX,
     &                       LUOUT)
              ENDIF
          ELSE
              TERMFLG = .TRUE.
              WRITE(LUOUT,1013) ROCFPM,VCLMOUT,THTMAX
          ENDIF
      ELSEIF(HAGL.GE.HCLMOUT)     THEN
C****     CONSTANT THETA TO VKEND OR HMAX AT THTFLY PITCH ATTITUDE
C****     HCLIMB AND HCLMOUT.
          IF(.NOT.INISEG2) THEN
C****         INITIATE CONSTANT PITCH ANGLE CLIMB AT SEGMENT TWO CLIMB
C****         GRADIENT.
              INISEG2  = .TRUE.
              THTMAX  = THTFLY
              LINENUM = LINENUM + 1
              WRITE(LUOUT,1012) HCLMOUT,TIME,GDIST,VKCAS,VKTAS
          ENDIF
          IF(ROCFPM.GE.ZERO) THEN
              IF      ((THETAF - THTFLY) .GE.  THTTOL) THEN
C****             REDUCE ALPHA
                  CALL PITCH('CLIMB ',ALPHA,-DADTCMD,DTIME,DTDTGEX,
     &                       LUOUT)
              ELSEIF ((THETAF - THTFLY) .LT. -THTTOL) THEN
C****             INCREASE ALPHA
                  CALL PITCH('CLIMB ',ALPHA, DADTCMD,DTIME,DTDTGEX,
     &                       LUOUT)
              ELSE
                  THETAF = THTFLY
              ENDIF
          ELSE
              TERMFLG = .TRUE.
              WRITE(LUOUT,1013) ROCFPM,VCLMOUT,THTMAX
          ENDIF
      ENDIF
```

```
1013     FORMAT(' SUBROUTINE TAKOFF IS UNABLE TO SIMULATE A CLIMB',               TAKOFF
   &            ' USING THE INPUTS OF CLIMB OUT SPEED OR PITCH'/,                 TAKOFF
   &            ' ATTITUDE BECAUSE THE RATE OF CLIMB IS LESS THAN',              TAKOFF
   &            ' ZERO.'/,' RATE OF CLIMB, CLIMB OUT SPEED, AND',                TAKOFF
   &            ' MAXIMUM PITCH ATTITUDE =',                                     TAKOFF
   &            F10.0, ' FEET/MINUTE',F10.0,' KNOTS',F10.0,' DEGREES.')          TAKOFF
C****                                                                            TAKOFF
C****    CHECK FOR TERMINATION CRITERIA.                                         TAKOFF
         IF(GDIST .GT.DISTMAX .OR. HAGL  .GE.HMAX   .OR.                         TAKOFF
   &        TIME  .GT.TIMEMAX .OR. VKCAS.GE.VKEND) TERMFLG = .TRUE.              TAKOFF
         IF(NCOUNT.EQ.10 .OR. TERMFLG) THEN                                      TAKOFF
C****       CALCULATE ADDITIONAL OUTPUT PARAMETERS, WRITE OUTPUT AND             TAKOFF
C****       INCREMENT LINE NUMBER COUNTER LINENUM.                               TAKOFF
            IF(.NOT.TERMFLG) THEN                                                TAKOFF
               DADT   = (ALPHA  - ALPHAJ)/FLOAT(DTIME)                           TAKOFF
               DTHTDT = (THETAF - THETAJ)/FLOAT(DTIME)                           TAKOFF
            ENDIF                                                                TAKOFF
            CALL FORCEX(ALPHA,CD,CL)                                             TAKOFF
            GAMMA  = ATAN(ROC/VTGS)*RX                                           TAKOFF
            ROCFPM = ROC*60.0                                                    TAKOFF
C****                                                                            TAKOFF
C****       RECALCULATE FLIGHTPATH ACCELERATION FOR OUTPUT.                      TAKOFF
            FPA = FPACCEL + G*SIN(GAMMAR)                                        TAKOFF
            IF(DTIME.GE.0.10D0) THEN                                             TAKOFF
               WRITE(LUOUT,1005) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,          TAKOFF
   &                             FPA,CL,CD,THETAF,ALPHA,GAMMA,DTHTDT,            TAKOFF
   &                             ROCFPM,XLF,THRUST,XENG                          TAKOFF
            ELSE                                                                 TAKOFF
               WRITE(LUOUT,1006) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,          TAKOFF
   &                             FPA,CL,CD,THETAF,ALPHA,GAMMA,DTHTDT,            TAKOFF
   &                             ROCFPM,XLF,THRUST,XENG                          TAKOFF
            ENDIF                                                                TAKOFF
            IF(TERMFLG) THEN                                                     TAKOFF
               WRITE(LUOUT,1014)                                                 TAKOFF
1014           FORMAT(/,' END OF TAKEOFF')                                       TAKOFF
               RETURN                                                            TAKOFF
            ENDIF                                                                TAKOFF
            LINENUM = LINENUM + 1                                                TAKOFF
            IF(LINENUM.GE.NPAGE) THEN                                            TAKOFF
C****          RESET LINE NUMBER COUNTER LINENUM AND WRITE HEADER FOR            TAKOFF
C****          NEW PAGE.                                                         TAKOFF
               LINENUM = 5                                                       TAKOFF
               WRITE(LUOUT,1010)                                                 TAKOFF
               WRITE(LUOUT,1004)                                                 TAKOFF
            ENDIF                                                                TAKOFF
C****       RESET INTEGRATION LOOP COUNTER ICOUNT TO 1 AND RESTART               TAKOFF
C****       THE AIRBORNE INTEGRATION LOOP.                                       TAKOFF
            ICOUNT = 1                                                           TAKOFF
            GO TO 29                                                             TAKOFF
         ENDIF                                                                   TAKOFF
  30  CONTINUE                                                                   TAKOFF
      END                                                                        TAKOFF
```

```
      SUBROUTINE LANDNG(AOA)                                                LANDNG   1
C**** THIS SUBROUTINE CONTROLS THE EXECUTION OF THE LANDING MANEUVER.      LANDNG   2
C**** LANDNG CALLS STEDYST TO OBTAIN THE REQUIRED VALUES OF THRUST          LANDNG   3
C**** THRUST AND ANGLE OF ATTACK FOR STEADY STATE APPROACH.  LANDNG         LANDNG   4
C**** CALLS FLARE TO EXECUTE THE FLARE AND CALLS ROLLS FOR THE GROUND       LANDNG   5
C**** PORTION OF THE LANDING.                                               LANDNG   6
C****                                                                      LANDNG   7
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                       CTRL   1
      PARAMETER (LUIN=3,LUOUT=4)                                              CTRL   2
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL   3
     &                 NEQ,NPAGE,TIME,TIMEROL                                 CTRL   4
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,  AIRCRFT   1
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                  AIRCRFT   2
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,        AERO   1
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                             AERO   2
      COMMON/ENGINE/   AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,       ENGINE   1
     &                 REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,         ENGINE   2
     &                 XENGOUT,XIDLE,XMIL,ZFN                               ENGINE   3
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ   AIRBORN   1
      COMMON/AIRSPED/  VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,   AIRSPED   1
     &                 VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                  AIRSPED   2
      COMMON/RUNWAY/   ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,       RUNWAY   1
     &                 HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,  RUNWAY   2
     &                 TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                  RUNWAY   3
      DOUBLEPRECISION  VTAS,DIST,ACCEL,VTASJ,RKGRND                          INTEG   1
      COMMON/INTEG/    VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                      INTEG   2
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG   1
     &                 ROC,RKAIR                                           FPINTEG   2
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG   3
     &                 ROC,RKAIR(40)                                       FPINTEG   4
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF ATMOS   1
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                       CONST   1
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                       CONST   2
      COMMON/CONST/    ASLSQR5,TWOOVR7                                       CONST   3
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,      FLAGS   1
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,        FLAGS   2
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,        FLAGS   3
     &          VFFLAG,WRITITR                                               FLAGS   4
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,       FLAGS   5
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,       FLAGS   6
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,   FLAGS   7
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                        FLAGS   8
      LOGICAL FINDV                                                         LANDNG  21
      REAL ATMARY(8),S(48)                                                  LANDNG  22
      EQUIVALENCE (TEMPR,ATMARY(1)),(FPVTAS,S(1))                           LANDNG  23
      DATA  SINKTD                                                          LANDNG  24
     &      /  10.0/                                                        LANDNG  25
```

```
      NAMELIST/LND/ALPHA,GAMMAPP,HFLARE,SINKTD,VKAPP                  LANDNG
      NAMELIST/LND2/ DTDTMX,SPLFLAG,WRITITR                           LANDNG
      ALPHA = AOA                                                     LANDNG
C****                                                                 LANDNG
C**** INPUT DATA LOADED INTO LANDNG THRU NAMELIST /LND/.               LANDNG
      READ(LUIN,LND)                                                  LANDNG
      READ(LUIN,LND2)                                                 LANDNG
C****                                                                 LANDNG
C**** TERMINATE PROGRAM IF APPROACH FLIGHT PATH ANGLE, FLARE HEIGHT OR LANDNG
C**** APPROACH SPEED NOT INITIALIZED.                                  LANDNG
      IF     (GAMMAPP.EQ. 1.0)    THEN                                LANDNG
             CALL HALT(LUIN,LUMSG,LUOUT,' GAMMAPP NOT INPUT BY USER.') LANDNG
      ELSEIF (GAMMAPP.GE.ZERO) THEN                                   LANDNG
             CALL HALT(LUIN,LUMSG,LUOUT,' GAMMAPP MUST BE LESS THAN ZERO.') LANDNG
      ENDIF                                                           LANDNG
      IF(HFLARE.EQ.-1.0) THEN                                         LANDNG
          CALL HALT(LUIN,LUMSG,LUOUT,' HFLARE NOT INPUT BY USER.')    LANDNG
      ELSEIF(VKAPP .EQ. 1.0 .AND. ALPHA.EQ. 99.0) THEN                LANDNG
     &    CALL HALT(LUIN,LUMSG,LUOUT,                                 LANDNG
     &                  ' VKAPP OR ALPHA NOT INPUT BY USER.')         LANDNG
      ENDIF                                                           LANDNG
      IF(ALPHA.EQ.99.0) THEN                                          LANDNG
C****     FIND ANGLE OF ATTACK FOR STEADY STATE APPROACH.             LANDNG
          FINDV  = .FALSE.                                            LANDNG
      ELSE                                                            LANDNG
C****     FIND AIRSPEED FOR STEADY STATE APPROACH; INITIALIZE CALIBRATED LANDNG
C****     AIRSPEED AT 100 KNOTS.                                      LANDNG
          VKAPP = 100.0                                               LANDNG
          FINDV  = .TRUE.                                             LANDNG
      ENDIF                                                           LANDNG
C****                                                                 LANDNG
C**** INITIALIZE CURVE FILES.                                          LANDNG
      CALL INICURV                                                    LANDNG
C****                                                                 LANDNG
C**** OBTAIN ATMOSPHERIC VARIABLES.                                    LANDNG
      IF(HCLEAR.GT.HFLARE) THEN                                       LANDNG
          HAGL = HCLEAR                                               LANDNG
      ELSE                                                            LANDNG
          HAGL = HFLARE                                               LANDNG
      ENDIF                                                           LANDNG
      PRESALT = HRUNWAY + HAGL                                        LANDNG
      CALL ATMOSPH(PRESALT,ATMARY)                                    LANDNG
C****                                                                 LANDNG
C**** INITIALIZE FLIGHT PATH ANGLE, TRUE AIRSPEED, MACH NUMBER AND QS. LANDNG
      VAPP      = VKAPP*FPSKTS                                        LANDNG
      FPVTAS    = VAPP/SQRT(SIGMA)                                    LANDNG
      GAMAPPR= GAMMAPP/RX                                             LANDNG
      GAMMAR  = GAMAPPR - ASIN((VWIND/FPVTAS)*SIN(GAMAPPR))           LANDNG
      CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS, VKTGS, LANDNG
                VTGS)                                                 LANDNG
      VKCAS   = VKAPP                                                 LANDNG
C****                                                                 LANDNG
C**** OBTAIN THRUST AND ANGLE OF ATTACK FOR STEADY STATE APPROACH.     LANDNG
      CALL STEDYST(ALPHA,FPVTAS,GAMMAR,FINDV)                         LANDNG
```

```
C****                                                                        LANDNG
C**** RECALCULATE APPROACH SPEED IF ANGLE OF ATTACK IS AN INPUT.             LANDNG
      IF(FINDV) THEN                                                         LANDNG
          VAPP  = FPVTAS*SQRT(SIGMA)                                         LANDNG
          VKAPP = VAPP/FPSKTS                                                LANDNG
      ENDIF                                                                  LANDNG
C****                                                                        LANDNG
C**** CALCULATE SINK RATE AT APPROACH IN FEET/MINUTE.                         LANDNG
      ROC    = FPVTAS*SIN(GAMMAR)                                            LANDNG
      SINKRT = -ROC*60.0                                                     LANDNG
      IF(ERRFLAG) THEN                                                       LANDNG
          TERMFLG = .TRUE.                                                   LANDNG
          AOA = ALPHA                                                        LANDNG
          WRITE(LUOUT,1001) VKAPP,GAMMAPP,PRESALT,ALPHA,ALPHMX,XENG,GWT      LANDNG
1001      FORMAT(' *** FAILED IN THRUST AND ANGLE OF ATTACK ON APPROACH',   LANDNG
  &              /,'  VKAPP,GAMMAPP,PRESALT =',3F10.2,                       LANDNG
  &              /,'  ALPHA,ALPHMX,XENG,GWT =',3F10.2,F10.0)                 LANDNG
          RETURN                                                             LANDNG
      ENDIF                                                                  LANDNG
C****                                                                        LANDNG
C**** DETERMINE LOAD FACTOR FOR FLARE.                                        LANDNG
      IF(HFLARE.GT.ZERO) CALL FLARENZ( ALPHA,GAMMAPP,GAMMATD,                LANDNG
  &                                   HAGL,HCLEAR,HFLARE,HRUNWAY,SINKTD,     LANDNG
  &                                   THRUST,VKAPP,XLFLARE,XLFMAX)           LANDNG
      WRITITR = .TRUE.                                                       LANDNG
C****                                                                        LANDNG
C**** WRITE HEADER INFORMATION FOR OUTPUT.                                    LANDNG
      LINENUM = 7                                                            LANDNG
      WRITE(LUOUT,1002) VKAPP   ,GAMMAPP  ,SINKRT     ,ALPHA    ,FLAP,       LANDNG
  &                     HRUNWAY,XLFMAX,                                      LANDNG
  &                              GAMMATD  ,SINKTD*60.0 ,THRUST   ,SPOILER,   LANDNG
  &                     DTEMPF  ,ALPHMX,                                     LANDNG
  &                     WNGLOD  ,GAMMARW               ,DADTCMD ,SPLFLAG,    LANDNG
  &                     VKWIND  ,DTDTMX                                      LANDNG
1002 FORMAT('1'   VKAPP  =',F7.1,      '  GAMMAPP =',F6.1,'  SINKRT =',F6.1, LANDNG
  &          '    ALPHA  =',F7.2,      '  FLAP    =',F6.1,'  HRUNWAY =',F7.0,  LANDNG
  &          '   XLFMAX  =',F6.2,      /,                                    LANDNG
  &          '           ',        '  GAMMATD =',F6.1,'  SINKTD =',F6.1,     LANDNG
  &          '   THRUST  =',F7.0,'  SPOILER =',F6.1,'  DTEMPF =',F7.1,       LANDNG
  &          '   ALPHMX  =',F6.2,      /,                                    LANDNG
  &          '  WNGLOD   =',F7.1,'  GAMMARW =',F6.1,'              ',        LANDNG
  &          '  DADTCMD  =',F7.2,'  SPLFLAG =',L6,   '  VKWIND =',F7.1,      LANDNG
  &          '  DTDTMX   =',F6.2,      /)                                    LANDNG
```

```
      IF(TERMFLG) THEN
C****      TERMINATE LANDING EXECUTION.
           AOA = ALPHA
           WRITE(LUOUT,1003)
1003       FORMAT(/,' END OF LANDING')
           RETURN
      ELSEIF(HFLARE.GE.HCLEAR) THEN
C****      CALCULATE VARIABLES FOR OUTPUT.
           OVERFLG = .TRUE.
           TIMF      = DBLE(ZERO)
C****
C****      WRITE HEADER RECORD FOR OUTPUT.
           WRITE(LUOUT,1004)
1004       FORMAT(' TIME  DIST.  WEIGHT  ALT.  VCAS  VTAS  VTGS',
   &              ' ACCEL    CL   CD THETA ALPHA GAMMA DTHET ',
   &              '  R/C LOAD  THRUST  XENG'/,
   &              ' (SEC) (FEET)  (LBS) (FEET) (KTS) (KTS) (KTS)',
   &              ' (FPS2)          (DEG) (DEG) (DEG)  /DT ',
   &              ' (FPM) FACT   (LBS) OR MU'/)
      ELSE
           STEADY = .TRUE.
           CALL APPROCH(ALPHA,GAMMAPP,GDIST,HAGL,HCLEAR,HFLARE,HRUNWAY)
      ENDIF
C****
C**** INITIALIZE APPROACH DISTANCE, APPDIST.
      APPDIST = GDIST
C****
C**** INITIALIZE LOAD FACTOR FOR FLARE.
      XLF       = XLFLARE
      XLFMAXJ = XLFMAX
      XLFMAX   = XLFLARE
C****
C**** CALL SUBROUTINE FLARE WITH FINAL LOAD FACTOR VALUE.
      IF(HFLARE.GT.ZERO) THEN
           STEADY    = .FALSE.
           LINENUM  = LINENUM + 1
           WRITE(LUOUT,1005) HFLARE,XLF
1005       FORMAT(' **** BEGIN FLARE AT ',F4.1,' FEET. ',
   &              '      LOAD FACTOR =',F8.4,' G''S ****')
           CALL FLARE(ALPHA,GDIST,ODIST,ROCTD,VKTGS)
      ENDIF
      XLFMAX = XLFMAXJ
      IF(TERMFLG) THEN
           AOA = ALPHA
           WRITE(LUOUT,1003)
           RETURN
      ENDIF
      CALL FORCEX(ALPHA,CD,CL)
      LINENUM = LINENUM + 2
      WRITE(LUOUT,1006)
1006  FORMAT(' **** TOUCHDOWN ****')
      CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,
                 VKTGS,VTGS)
```

```
      AIRDIST  = GDIST
      FLRDIST = AIRDIST - APPDIST
      GAMMA   = ATAN(ROC/VTGS)*RX
      SINKRT   = -SINKTD*60.0
      THETAF  = ALPHA
      IF(DTIME.GE.0.10D0) THEN
          WRITE(LUOUT,1007) TIME,GDIST,GWT,ZERO,VKCAS,VKTAS,VKTGS,
     &                        FPACCEL,CL,CD,THETAF,ALPHA,GAMMA,DTDT,-SINKRT,
     &                        XLF,THRUST,XMU
 1007     FORMAT(F6.1,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,
     &              F6.2,F9.0,F7.3)
      ELSE
          WRITE(LUOUT,1008) TIME,GDIST,GWT,ZERO,VKCAS,VKTAS,VKTGS,FPACCEL,CL,
     &                        FPACCEL,CL,CD,THETAF,ALPHA,GAMMA,DTDT,-SINKRT,
     &                        XLF,THRUST,XMU
 1008     FORMAT(F6.2,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,
     &              F6.2,F9.0,F7.3)
      ENDIF
C****
C**** INITIALIZE GROUND ROLL INTEGRATION VARIABLES WITH PREVIOUS
C**** AIRBORNE INTEGRATION VARIABLES FOR SUBROUTINE ROLL.
      TIMEFLD = FLOAT(TIME)
      VTAS      = FPVTAS
      CALL ROLL(ALPHA,GDIST)
      AOA       = ALPHA
      WRITE(LUOUT,1009) INT(HCLEAR),GDIST-ODIST,AIRDIST,GDIST-AIRDIST,
     &                   APPDIST,FLRDIST,ABARG
 1009 FORMAT(/' DISTANCE OVER A',I4,' FOOT OBSTACLE =',F7.1,' FEET; ',
     &       ' AIR DISTANCE   =',F7.1,' FEET; GROUND ROLL DISTANCE =',
     &       F7.1,' FEET;'/,
     &       '           PRE-FLARE DISTANCE =',F7.1,' FEET; ',
     &       ' FLARE DISTANCE =',F7.1,' FEET; AVERAGE DECELERATION =',
     &       F6.3,' G'/,)
      RETURN
      END
```

Subroutine STEDYST

```
      SUBROUTINE STEDYST(ALPHA,FPVTAS,GAMMAR,FINDV)                           STEDYST    1
C**** THIS SUBROUTINE CALCULATES THE REQUIRED VALUES OF THRUST AND             STEDYST    2
C**** ANGLE OF ATTACK OR THRUST AND AIRSPEED FOR ZERO ACCELERATION ALONG       STEDYST    3
C**** AND NORMAL TO THE FLIGHT PATH, DV/DT AND DG/DT, RESPECTIVELY.  THE       STEDYST    4
C**** OUTER LOOP VARIES THRUST, WHILE THE TWO INNER LOOPS VARY                 STEDYST    5
C**** ALPHA FROM ALPHMN TO ALPHMX OR FPVTAS FROM VTASMN TO VTASMX.  FOR        STEDYST    6
C**** A FIXED VALUE OF THRUST (OUTER LOOP), FUNCTIONS DVDT (ACCELERATION       STEDYST    7
C**** ALONG FLIGHT PATH), AND DGDT (ACCELERATION NORMAL TO FLIGHT PATH),       STEDYST    8
C**** ARE CALLED WITH THE VALUES OF ALPHA (INNER LOOP).  THESE TWO             STEDYST    9
C**** FUNCTION SUBROUTINES CALL FORCEX.  SUBROUTINES ITRLND AND ZEROX          STEDYST   10
C**** ARE BOTH ZERO FINDERS.                                                   STEDYST   11
C****                                                                          STEDYST   12
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                           CTRL    1
      PARAMETER (LUIN=3,LUOUT=4)                                                  CTRL    2
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,      CTRL    3
     &                 NEQ,NPAGE,TIME,TIMEROL                                     CTRL    4
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX.FLT,GWT,HZ,      AIRCRFT    1
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                      AIRCRFT    2
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,            AERO    1
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                                 AERO    2
      COMMON/ENGINE/   AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,           ENGINE    1
     &                 REVNDX,RC.THRUST,VTANGLE,WFUEL,XENG,XENGFLD,             ENGINE    2
     &                 XENGOUT,XIDLE,XMIL,ZFN                                   ENGINE    3
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF     ATMOS    1
      PARAMETER (ASL=661.48,FPSKTS=1.687806.G= 32.174,                           CONST    1
     &            RX=57.29577951308,TSLF=59.0,ZERO=0.0)                          CONST    2
      COMMON/CONST/    ASLSQR5,TWOOVR7                                           CONST    3
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,          FLAGS    1
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,            FLAGS    2
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,            FLAGS    3
     &          VFFLAG,WRITITR                                                   FLAGS    4
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,           FLAGS    5
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,           FLAGS    6
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,       FLAGS    7
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                            FLAGS    8
      LOGICAL FINDV                                                            STEDYST   20
      REAL INTERP                                                              STEDYST   21
      EXTERNAL DGDT,DVDT                                                       STEDYST   22
      DATA   ALPHA0,ALPHA4, DALPHA, DVTAS,    EPS,  FACTOR, FNSTEP,MAXITER     STEDYST   23
     &         /  0.0,     4.0,      1.0,     1.0, 0.0001,    0.95,    1.05,        31 /   STEDYST   24
      ERRFLAG  = .FALSE.                                                       STEDYST   25
C****                                                                          STEDYST   26
C**** INITIALIZE NET THRUST TOLERANCE TO THE GREATER OF 0.1% OF GROSS          STEDYST   27
C**** WEIGHT OR 5 POUNDS.                                                      STEDYST   28
      IF(GWT.GT.5000.) THEN                                                    STEDYST   29
          TOLRNCE = 0.001*GWT                                                  STEDYST   30
      ELSE                                                                     STEDYST   31
          TOLRNCE = 5.0                                                        STEDYST   32
      ENDIF                                                                    STEDYST   33
```

```
C****
C**** INITIALIZE UPPER AND LOWER SEARCH BOUNDARIES FOR THRUST TO
C**** THE QUOTIENT OF GROSS WEIGHT OVER THE ENGINE MULTIPLICATIVE FACTOR
C**** AND 1% OF THE GROSS WEIGHT, RESPECTIVELY.
      BOUNDU = GWT/XENG
      BOUNDL = GWT/100.0
C****
C**** MAKE FIRST GUESS AT ALPHA OR VKAPP.
      IF(.NOT.FINDV) THEN
C****       FIND ANGLE OF ATTACK.
            CALL FORCEX(ALPHA4,CD4,CL4)
            CALL FORCEX(ALPHA0,CD0,CL0)
            ALPHA    = INTERP(ALPHA4,ALPHA0,CL4,CL0,GWT/QS)
            CALL FORCEX(ALPHA ,CD ,CL)
            ALPHMX  = ALPHA + 2.5
            ALPHMN  = -ALPHMX
      ELSE
C****       FIND AIRSPEED.
C****
C****       FIND APPROXIMATE CL AND CD FOR MACH NUMBER AT 100 KNOTS
C****       AIRSPEED.
            CALL FORCEX(ALPHA ,CD   ,CL)
C****
C****       CALCULATE FLIGHT PATH TRUE AIRSPEED BASED ON THE APPROXIMATED
C****       CL AND ASSUMPTION OF LIFT EQUAL TO WEIGHT AND ALPHA EQUAL TO
C****       ZERO.
            FPVTAS  = SQRT(2.0*GWT*COS(GAMMAR)/(CL*RHO*SWING))
            CALL SPEED( GAMMAR,FPVTAS,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                        VKTGS,VTGS)
            CALL FORCEX(ALPHA ,CD   ,CL)
            VTASMX  = FPVTAS + 10.0
            VTASMN  = FPVTAS - 10.0
      ENDIF
C****
C**** MAKE FIRST GUESS AT THRUST.
      TFIRST = 1.06*(CD*QS + GWT*SIN(GAMMAR))
      IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
   &        WRITE(LUOUT,9001) TFIRST,ALPHA,VKCAS,CD,CL,QS
9001 FORMAT('**** INITIAL CONDITIONS ****'/,
   &              ' THRUST, ALPHA,  VKAPP,    CD,     CL,      QS ='/,
   &              F7.0, F7.2,   F9.1,        2F8.4,    F10.1/)
   19 LABLNO  = 19
      ERROR   = 99.
      JFLAG   = 0
      THRUST  = TFIRST
      IF(FINDV) THEN
            FG = THRUST/COSD(ALPHA+AIT)
            IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
   &              WRITE(LUOUT,9002) ICOUNT,LABLNO,THRUST,FPVTAS,DVDTHI,
   &                                        JFLAG,ERROR,ERRORJ
```

```
      DO 10 ICOUNT=1,MAXITER
         IF(JFLAG.LE.1) ERRORJ = ERROR
C****
C****    FIND VALUE OF THRUST SUCH THAT DV/DT IS NEGATIVE AT FPVTAS
C****    EQUAL TO VTASMX, WITH A REDUCTION IN FPVTAS RESULTING IN
C****    LESS DRAG DV/DT WILL CHANGE SIGN(I.E. A BOUNDED INTERVAL IN
C****    WHICH DV/DT EQUAL TO ZERO WILL BE FOUND).
         VTASHI  = VTASMX
         CALL SPEED( GAMMAR,VTASHI,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                     VKTGS,VTGS)
         DVDTHI  = DVDT(ALPHA,VTASHI,GAMMAR,GWT)
         IF (DVDTHI.GE.0.0) THEN
            TFIRST = TFIRST/FNSTEP
            IF(TFIRST.GE.BOUNDL) GO TO 19
            GO TO 199
         ENDIF
C****
C****    SEARCH FOR DV/DT EQUAL TO ZERO FROM VTASMX TO VTASMN.
   29    LABLNO = 29
         VTASLO = VTASHI - DVTAS
         FG     = THRUST/COSD(ALPHA+AIT)
         IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
    &    WRITE(LUOUT,9002) ICOUNT,LABLNO,THRUST,VTASLO,DVDTHI,
    &                      JFLAG,ERROR,ERRORJ
         IF(VTASLO.LT.VTASMN) THEN
            ERRFLAG = .TRUE.
            WRITE(LUOUT,1001)THRUST,VTASLO,ERROR,ERRORJ,ICOUNT
 1001       FORMAT(' *** FAILED IN STEDYST.  NO SOLUTION FOR DVDT.',
    &              /,' THRUST,FPVTAS,ERROR,ERRORJ,ICOUNT =',
    &              2F9.2,2F10.4,I3)
            RETURN
         ENDIF
         CALL SPEED( GAMMAR,VTASLO,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                     VKTGS,VTGS)
         DVDTLO= DVDT(ALPHA,VTASLO,GAMMAR,GWT)
C****
C****    CHECK TO SEE IF DV/DT EQUAL TO ZERO IS BOUNDED.
         IF (DVDTLO*DVDTHI.GT.0.0) THEN
            VTASHI  = VTASLO
            DVDTHI  = DVDTLO
            GO TO 29
         ENDIF
```

```
C****
C****         ONCE FPVTAS INTERVAL FOR DV/DT EQUAL TO ZERO IS BOUNDED,
C****         CALL FUNCTION ZEROX TO FIND FPVTAS1, THE VALUE OF FPVTAS FOR
C****         DV/DT EQUALS ZERO.
              FPVTAS1 = ZEROX(VTASLO,VTASHI,DVDT,ALPHA,GAMMAR,GWT,EPS,
     &                        FINDV)
C****
C****         SEARCH FOR DG/DT SAME AS ABOVE SEARCH FOR DV/DT EQUALS ZERO.
C****         IF DG/DT EQUALS ZERO AT FPVTAS = VTASMX, PROGRAM FAILS.
C****         FPVTAS2 IS REQUIRED VALUE OF FPVTAS FOR DG/DT EQUAL TO ZERO.
              VTASHI  = VTASMX
              CALL SPEED( GAMMAR,VTASHI,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                          VKTGS,VTGS)
              DGDTHI = DGDT(ALPHA,VTASHI,GAMMAR,GWT)
              IF(DGDTHI.LT.0.0) GO TO 199
   39         LABLNO = 39
              VTASLO = VTASHI - DVTAS
              FG      = THRUST/COSD(ALPHA+AIT)
              IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
     &           WRITE(LUOUT,9003)  ICOUNT,LABLNO,THRUST,VTASLO,DGDTHI,
     &                              JFLAG,ERROR,ERRORJ
              IF(VTASLO.LT.VTASMN) THEN
                 TFIRST = TFIRST/FNSTEP
                 IF(TFIRST.GE.BOUNDL) GO TO 19
                 GO TO 199
              ENDIF
              CALL SPEED( GAMMAR,VTASLO,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                          VKTGS,VTGS)
              DGDTLO  = DGDT(ALPHA,VTASLO,GAMMAR,GWT)
              IF(DGDTLO*DGDTHI.GT.0.0) THEN
                 VTASHI  = VTASLO
                 DGDTHI = DGDTLO
                 GO TO 39
              ENDIF
              FPVTAS2   = ZEROX(VTASLO,VTASHI,DGDT,ALPHA,GAMMAR,GWT,EPS,
     &                          FINDV)
```

```
C****
C****       FIND ERROR BETWEEN FPVTAS1 AND FPVTAS2.
            ERROR = (FPVTAS1 - FPVTAS2)/VTASMX
            IF    ((ABS(ERROR).LT.0.001                          ).OR.
     &              (ABS(ERROR).LT.0.005 .AND. ICOUNT.GT.25).OR.
     &              (JFLAG.EQ.3                                  )) THEN
               FPVTAS = FPVTAS1
               CALL SPEED(GAMMAR,FPVTAS,0.0,AMACH,QS,VKCAS,VKEAS,VKTAS,
                          VKTGS,VTGS)
               RETURN
            ELSE
C****          MAKE NEW GUESS AT THRUST AND REPEAT DV/DT AND DG/DT LOOPS
C****          UNTIL FPVTAS1 EQUALS FPVTAS2 (WITHIN TOLERANCES).
               CALL ITRLND(ERROR,ERRORJ,THRUST,FACTOR,TOLRNCE,JFLAG)
            ENDIF
   10    CONTINUE
  199    WRITE(LUOUT,1003) TFIRST,DGDTHI,ERROR,ERRORJ,FPVTAS1,FPVTAS2,
     &                 ICOUNT
 1003    FORMAT(' *** FAILED IN SUBROUTINE STEDYST.',/,
     &          ' TFIRST,  DGDTHI,   ERROR,   ERRORJ,  FPVTAS1,',
     &          ' FPVTAS2,ICOUNT = ',/,F8.0,5F10.4,I7)
```

```
      ELSE
         FG    = THRUST/COSD(ALPHHI+AIT)
         IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
     &      WRITE(LUOUT,9004) ICOUNT,LABLNO,THRUST,ALPHA,DVDTHI,
     &                        JFLAG,ERROR,ERRORJ
         DO 20 ICOUNT=1,MAXITER
            IF(JFLAG.LE.1) ERRORJ = ERROR
C****
C****       FIND VALUE OF THRUST SUCH THAT DV/DT IS NEGATIVE AT ALPHA
C****       EQUAL TO ALPHMX, WITH A REDUCTION IN ALPHA RESULTING IN LESS
C****       LESS DRAG DV/DT WILL CHANGE SIGN(I.E. A BOUNDED INTERVAL IN
C****       WHICH DV/DT EQUAL TO ZERO WILL BE FOUND).
            ALPHHI  = ALPHMX
            DVDTHI = DVDT(ALPHHI,FPVTAS,GAMMAR,GWT)
            IF (DVDTHI.GE.0.0) THEN
               TFIRST = TFIRST/FNSTEP
               IF(TFIRST.GE.BOUNDL) GO TO 19
               GO TO 299
            ENDIF
C****
C****       SEARCH FOR DV/DT EQUAL TO ZERO FROM ALPHMX TO -ALPHMX.
 49         LABLNO = 49
            ALPHLO = ALPHHI - DALPHA
            FG       = THRUST/COSD(ALPHLO+AIT)
            IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
     &         WRITE(LUOUT,9004)  ICOUNT,LABLNO,THRUST,ALPHLO,DVDTHI,
     &                            JFLAG,ERROR,ERRORJ
            IF(ALPHLO.LT.ALPHMN) THEN
               ERRFLAG = .TRUE.
               WRITE(LUOUT,1002)THRUST,ALPHLO,ERROR,ERRORJ,ICOUNT
1002           FORMAT( ' *** FAILED IN STEDYST.  NO SOLUTION FOR DVDT.',
     &                /,' THRUST,ALPHA,ERROR,ERRORJ,ICOUNT =',
     &                2F9.2,2F10.4,I3)
               RETURN
            ENDIF
            DVDTLO = DVDT(ALPHLO,FPVTAS,GAMMAR,GWT)
C****
C****       CHECK TO SEE IF DV/DT EQUAL TO ZERO IS BOUNDED.
            IF (DVDTLO*DVDTHI.GT.0.0) THEN
               ALPHHI  = ALPHLO
               DVDTHI  = DVDTLO
               GO TO 49
            ENDIF
C****
C****       ONCE ALPHA INTERVAL FOR DV/DT EQUAL TO ZERO IS BOUNDED,
C****       CALL FUNCTION ZEROX TO FIND ALPHA1, THE VALUE OF ALPHA FOR
C****       DV/DT EQUALS ZERO.
            ALPHA1 = ZEROX(ALPHLO,ALPHHI,DVDT,FPVTAS,GAMMAR,GWT,EPS,
     &                     FINDV)
```

```
C****
C****          SEARCH FOR DG/DT SAME AS ABOVE SEARCH FOR DV/DT EQUALS ZERO.
C****          IF DG/DT EQUALS ZERO AT ALPHA = ALPHMX, PROGRAM FAILS.
C****          ALPHA2 IS REQUIRED VALUE OF ALPHA FOR DG/DT EQUAL TO ZERO.
               ALPHHI = ALPHMX
               DGDTHI = DGDT(ALPHHI,FPVTAS,GAMMAR,GWT)
               IF(DGDTHI.LT.0.0) GO TO 299
   59          LABLNO = 59
               ALPHLO = ALPHHI - DALPHA
               FG     = THRUST/COSD(ALPHLO+AIT)
               IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999)
     &            WRITE(LUOUT,9005) ICOUNT,LABLNO,THRUST,ALPHLO,DGDTHI,
     &                              JFLAG,ERROR,ERRORJ
               IF(ALPHLO.LT.ALPHMN) THEN
                  TFIRST = TFIRST/FNSTEP
                  IF(TFIRST.GE.BOUNDL) GO TO 19
                  GO TO 299
               ENDIF
               DGDTLO = DGDT(ALPHLO,FPVTAS,GAMMAR,GWT)
               IF(DGDTLO*DGDTHI.GT.0.0) THEN
                  ALPHHI = ALPHLO
                  DGDTHI = DGDTLO
                  GO TO 59
               ENDIF
               ALPHA2 = ZEROX(ALPHLO,ALPHHI,DGDT,FPVTAS,GAMMAR,GWT,EPS,
     &                        FINDV)
C****
C****          FIND ERROR BETWEEN ALPHA1 AND ALPHA2.
               ERROR = (ALPHA1 - ALPHA2)/ALPHMX
               IF   ((ABS(ERROR).LT.0.005                     ).OR.
     &              (ABS(ERROR).LT.0.010 .AND. ICOUNT.GT.25).OR.
     &              (JFLAG.EQ.3                               )) THEN
                  ALPHA = ALPHA1
                  RETURN
               ELSE
C****          MAKE NEW GUESS AT THRUST AND REPEAT DV/DT AND DG/DT LOOPS
C****          UNTIL ALPHA1 EQUALS ALPHA2 (WITHIN TOLERANCES).
               CALL ITRLND(ERROR,ERRORJ,THRUST,FACTOR,TOLRNCE,JFLAG)
               ENDIF
   20       CONTINUE
  299       WRITE(LUOUT,1004) TFIRST,DGDTHI,ERROR,ERRORJ,ALPHA1,ALPHA2,
     &                        ICOUNT
 1004       FORMAT(' *** FAILED IN SUBROUTINE STEDYST.'/,
     &             ' TFIRST,  DGDTHI,  ERROR,  ERRORJ,  ALPHA1,',
     &             '  ALPHA2,ICOUNT = '/,F8.0,5F10.4,I7)
         ENDIF
```

```
      ERRFLAG = .TRUE.                                                STEDYST
 9002 FORMAT(' ICOUNT, LABLNO, THRUST, FPVTAS,  DVDT, JFLAG,',        STEDYST
     &       ' ERROR, ERRORJ = ',/,                                   STEDYST
     &       2I8,  F8.0,  F8.1,  F8.4,   I7, 2F8.3)                   STEDYST
 9003 FORMAT(' ICOUNT, LABLNO, THRUST, FPVTAS,  DGDT, JFLAG,',        STEDYST
     &       ' ERROR, ERRORJ = ',/,                                   STEDYST
     &       2I8,  F8.0,  F8.1,  F8.4,   I7, 2F8.3)                   STEDYST
 9004 FORMAT(' ICOUNT, LABLNO, THRUST,  ALPHA,  DVDT, JFLAG,',        STEDYST
     &       ' ERROR, ERRORJ = ',/,                                   STEDYST
     &       2I8,  F8.0,  F8.1,  F8.4,   I7, 2F8.3)                   STEDYST
 9005 FORMAT(' ICOUNT, LABLNO, THRUST,  ALPHA,  DGDT, JFLAG,',        STEDYST
     &       ' ERROR, ERRORJ = ',/,                                   STEDYST
     &       2I8,  F8.0,  F8.1,  F8.4,   I7, 2F8.3)                   STEDYST
      RETURN                                                          STEDYST
      END                                                             STEDYST
```

Subroutine FLARENZ

```
      SUBROUTINE FLARENZ(ALPHA,GAMMAPP,GAMMATD,HAGL,HCLEAR,HFLARE,           FLARENZ    1
     &                  HRUNWAY,SINKTD,THRUST,VKAPP,XLFLARE,XLFMAX)           FLARENZ    2
C**** THIS SUBROUTINE DETERMINES THE CORRECT TARGET LOAD FACTOR FOR            FLARENZ    3
C**** THE LANDING FLARE.                                                       FLARENZ    4
C****                                                                          FLARENZ    5
C**** SUBROUTINE ITRLND IS A ZERO-FINDING ROUTINE, WHICH VARIES THE            FLARENZ    6
C**** INDEPENDENT VARIABLE BASED ON THE SIZE AND SIGN OF THE ERROR.            FLARENZ    7
C**** THE LOOP CONTAINING ITRLND IS EXITED SUCCESSFULLY WHEN JFLAG = 3,        FLARENZ    8
C**** OR THE MAGNITUDE OF THE ERROR IS LESS THAN SOME ACCEPTABLE VALUE.        FLARENZ    9
C**** VARIABLE JCOUNT STORES THE NUMBER OF ITERATIONS WHICH IS LIMITED         FLARENZ   10
C**** TO 25.                                                                   FLARENZ   11
C****                                                                          FLARENZ   12
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                           CTRL    1
      PARAMETER (LUIN=3,LUOUT=4)                                                  CTRL    2
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,       CTRL    3
     &                NEQ,NPAGE,TIME,TIMEROL                                      CTRL    4
      DOUBLEPRECISION FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,         FPINTEG    1
     &                ROC,RKAIR                                                FPINTEG    2
      COMMON/FPINTEG/ FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,         FPINTEG    3
     &                ROC,RKAIR(40)                                            FPINTEG    4
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                          CONST    1
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                          CONST    2
      COMMON/CONST/   ASLSQR5,TWOOVR7                                           CONST    3
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,          FLAGS    1
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,            FLAGS    2
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,            FLAGS    3
     &         VFFLAG,WRITITR                                                   FLAGS    4
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,           FLAGS    5
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,           FLAGS    6
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,       FLAGS    7
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                            FLAGS    8
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,           CHARV    1
     &          THRCRV*3,TKOTYPE*7                                              CHARV    2
      COMMON/CHARV/   ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,          CHARV    3
     &                TKOTYPE                                                   CHARV    4
      REAL INTERP                                                             FLARENZ   17
      DATA  EPSROC,FACTOR, MAXITER, TOLRNCE                                   FLARENZ   18
     &    /   0.5,    1.03,      25,  0.00001/                                FLARENZ   19
C****                                                                         FLARENZ   20
C**** CALCULATE INITIAL LOAD FACTOR GUESS FOR FLARE.                          FLARENZ   21
      VAPP     = VKAPP*FPSKTS                                                 FLARENZ   22
      GAMAPPR = GAMMAPP/RX                                                    FLARENZ   23
      GAMATDR = ASIN(-SINKTD/VAPP)                                            FLARENZ   24
      XLFLARE  = (VAPP**2*(COS(GAMATDR) - COS(GAMAPPR))/                      FLARENZ   25
     &           (G*HFLARE)) + 1.0                                            FLARENZ   26
      GAMMATD = GAMATDR*RX                                                    FLARENZ   27
      IF(XLFLARE.GT.XLFMAX)   CALL HALT(LUIN.LUMSG,LUOUT,                     FLARENZ   28
     &                        ' FLARE LOAD FACTOR EXCEEDS MAXIMUM LIMIT.')    FLARENZ   29
```

```
C****                                                                   FLARENZ
C**** INITIALIZE PARAMETERS FOR FLARE ITERATIONS.                       FLARENZ
      AOAAPP   = ALPHA                                                  FLARENZ
      BOUNDU   = XLFMAX                                                 FLARENZ
      ERROR    = 1.0                                                    FLARENZ
      FNAPP    = THRUST                                                 FLARENZ
      FPVTASJ  = FPVTAS                                                 FLARENZ
      JFLAG    = 0                                                      FLARENZ
      IF(.NOT.WRITITR) THEN                                             FLARENZ
         IF(JDEBUG.EQ.6666 .OR. JDEBUG.EQ.9999) THEN                    FLARENZ
            WRITE(LUOUT,1001)                                           FLARENZ
         ELSE                                                           FLARENZ
            WRITE(LUOUT,1002)                                           FLARENZ
         ENDIF                                                          FLARENZ
      ENDIF                                                             FLARENZ
      DO 10 JCOUNT=1,MAXITER+1                                          FLARENZ
         IF (JFLAG.NE.4) THEN                                           FLARENZ
            IF (JFLAG.LE.1) ERRORJ = ERROR                              FLARENZ
            ALPHA   = AOAAPP                                            FLARENZ
            DTIME   = DTIMEJ                                            FLARENZ
            FDIST   = ZERO                                              FLARENZ
            FPVTAS  = FPVTASJ                                           FLARENZ
            ROCTDJ  = ROCTD                                             FLARENZ
            THRUST  = FNAPP                                             FLARENZ
            TIME    = DBLE(ZERO)                                        FLARENZ
            XLF     = XLFLARE                                           FLARENZ
            XLFMAX  = XLFLARE                                           FLARENZ
            LINENUM = 6                                                 FLARENZ
            IF(WRITITR) WRITE(LUOUT,1003) XLF,SINKTD,FACTOR,ERROR,JFLAG, FLARENZ
                                          JCOUNT                        FLARENZ
            IF(HCLEAR.LE.HFLARE)THEN                                    FLARENZ
               IF(WRITITR) WRITE(LUOUT,1004)                            FLARENZ
            ELSE                                                        FLARENZ
               STEADY = .TRUE.                                          FLARENZ
               CALL APPROCH(ALPHA,GAMMAPP,FDIST,HAGL,HCLEAR,HFLARE,     FLARENZ
     &                          HRUNWAY)                                FLARENZ
            ENDIF                                                       FLARENZ
            STEADY = .FALSE.                                            FLARENZ
            CALL FLARE(ALPHA,FDIST,ODIST,ROCTD,VKTGS)                   FLARENZ
            IF(TERMFLG) RETURN                                          FLARENZ
            WRITE(LUOUT,1005) -ROCTD*60.0,JFLAG,JCOUNT                  FLARENZ
            DELTROC = -ROCTD - SINKTD                                   FLARENZ
            IF(SINKTD.NE.ZERO) THEN                                     FLARENZ
               ERROR = DELTROC/SINKTD                                   FLARENZ
            ELSE                                                        FLARENZ
               ERROR = DELTROC                                          FLARENZ
            ENDIF                                                       FLARENZ
```

```
                IF (JFLAG.LT.3 .AND. ABS(DELTROC).GE.EPSROC/60.0) THEN
                   XLFJ = XLF
C****
C****              GUESS NEW LOAD FACTOR BASED ON ERROR--CALL ITRLND.
                   CALL ITRLND(ERROR,ERRORJ,XLFLARE,FACTOR,TOLRNCE,JFLAG)
                   IF(JCOUNT.GT.25    .OR.
     &                XLFLARE.GT.BOUNDU .OR. XLFLARE.LE.1.0) THEN
C****                 RESET PARAMETERS TO PRE-FLARE SETTINGS.
                      ALPHA  = AOAAPP
                      DTIME  = DTIMEJ
                      FPVTAS = FPVTASJ
                      THRUST = FNAPP
                      XLF    = XLFLARE
                      XLFMAX = BOUNDU
                      WRITE(LUOUT,1006)XLFLARE,ROCTD,SINKTD,ERROR,ERRORJ,
     &                                 JCOUNT,JFLAG
                      RETURN
                   ENDIF
                ELSE
                   JFLAG = 4
                ENDIF
             ENDIF
   10 CONTINUE
 1001 FORMAT('1' ** LANDING **',/)
 1002 FORMAT(  ' ** LANDING **',/)
 1003 FORMAT('1' XLFLARE, SINKTD, FACTOR, ERROR, JFLAG, JCOUNT=',/,
     &         F11.4,          2F8.2,   F8.4,          2I8  ,/)
 1004 FORMAT(' TIME  DIST.  WEIGHT   ALT.  VCAS  VTAS  VTGS',
     &        ' ACCEL    CL    CD  THETA  ALPHA  GAMMA  DTHET ',
     &        '  R/C  LOAD  THRUST  XENG',/,
     &        '(SEC)  (FEET)   (LBS)  (FEET)  (KTS)  (KTS)  (KTS)',
     &        '(FPS2)             (DEG)  (DEG)  (DEG)   /DT ',
     &        '(FPM)  FACT   (LBS)  OR MU',/)
 1005 FORMAT(' *** SINK RATE AT TOUCHDOWN =',F7.1,' FEET/MINUTE',
     &        '   CONVERGENCE FLAG =',I3,
     &        '   FLARENZ ITERATION NUMBER =',I3,' ***')
 1006 FORMAT(' *** FAILED IN FLARENZ',/,
     &        '   XLFLARE,ROCTD,SINKTD =',F8.4,2F9.2,/,
     &        '   ERROR,ERRORJ,JCOUNT,JFLAG =',2F7.4,2I3)
C****
C**** TOUCHDOWN RATE WITHIN TOLERANCE; RESET PARAMETERS TO PRE-FLARE
C**** SETTINGS.
      ALPHA  = AOAAPP
      DTIME  = DTIMEJ
      FPVTAS = FPVTASJ
      THRUST = FNAPP
      XLF    = XLFLARE
      XLFMAX = BOUNDU
      IF(JCOUNT.GE.1) THEN
         XLFLARE = (INTERP(XLF,XLFJ,ROCTD,ROCTDJ,-SINKTD) + 2.0*XLF)/3.0
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE APPROCH(ALPHA,GAMMAPP,GDIST,HAGL,HCLEAR,HFLARE,HRUNWAY)    APPROCH
C**** THIS SUBROUTINE CONTROLS THE EXECUTION OF THE LANDING APPROACH       APPROCH
C**** WHEN THE OBSTACLE CLEARANCE HEIGHT, HCLEAR, IS ABOVE THE FLARE       APPROCH
C**** HEIGHT, HFLARE.  USING THE VALUES FROM SUBROUTINE STEDYST AS         APPROCH
C**** CONSTANTS, APPROCH DOES NOT USE RUNGE-KUTTA NUMERICAL INTEGRATION.   APPROCH
C****                                                                      APPROCH
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                    CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                           CTRL
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL
     &                 NEQ,NPAGE,TIME,TIMEROL                              CTRL
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,  AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                  AIRCRFT
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,     AERO
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                          AERO
      COMMON/ENGINE/   AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,      ENGINE
     &                 REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD         ENGINE
     &                 XENGOUT,XIDLE,XMIL,ZFN                              ENGINE
      COMMON/AIRSPED/  VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,   AIRSPED
     &                 VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                  AIRSPED
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG
     &                 ROC,RKAIR                                           FPINTEG
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,    FPINTEG
     &                 ROC,RKAIR(40)                                       FPINTEG
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF  ATMOS
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                     CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                     CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                     CONST
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,     FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,       FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,       FLAGS
     &         VFFLAG,WRITITR                                              FLAGS
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,     FLAGS
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,     FLAGS
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                      FLAGS
      REAL ATMARY(8),S(48),INTERP                                          APPROCH
      EQUIVALENCE (TEMPR,ATMARY(1)),(FPVTAS,S(1))                          APPROCH
      DATA APPDTIM/0.5/                                                    APPROCH
C****                                                                      APPROCH
C**** WRITE HEADER RECORD FOR OUTPUT.                                      APPROCH
      IF(WRITITR) WRITE(LUOUT,1001)                                        APPROCH
 1001 FORMAT(' TIME  DIST.  WEIGHT   ALT.  VCAS  VTAS  VTGS',              APPROCH
     &       ' ACCEL    CL    CD THETA ALPHA GAMMA DTHET ',                APPROCH
     &       '  R/C LOAD  THRUST  VENG'/,                                  APPROCH
     &       '(SEC) (FEET)  (L         .) (KTS) (KTS) (KTS)',              APPROCH
     &       '(FPS2)          (DEG) (DEG) (DEG)  /DT ',                    APPROCH
     &       '(FPM) FACT  (LBS) OR MU'/)                                   APPROCH
      GAMAPPR = GAMMAPP/RX                                                 APPROCH
      GAMMAR  = GAMAPPR - ASIN((VWIND/FPVTAS)*SIN(GAMAPPR))                APPROCH
      VAPP    = VKAPP*FPSKTS                                               APPROCH
```

```
C****                                                                          APPROCH
C**** CALCULATE SINK RATE AT APPROACH IN FEET/MINUTE.                          APPROCH
      ROC     = FPVTAS*SIN(GAMMAR)                                             APPROCH
      SINKRT = -ROC*60.0                                                       APPROCH
C****                                                                          APPROCH
C**** CALCULATE VARIABLES FOR OUTPUT ON APPROACH.                               APPROCH
      DELTAH   = -ROC*APPDTIM                                                  APPROCH
      DGDIST   = -DELTAH/TAN(GAMAPPR) - VWIND*APPDTIM                          APPROCH
      DTDT     = FPACCEL = ZERO                                                APPROCH
      GDIST    = -DGDIST                                                       APPROCH
      HAGL     = HCLEAR + DELTAH                                               APPROCH
      OVERFLG = .FALSE.                                                        APPROCH
      TIME     = DBLE(-APPDTIM)                                                APPROCH
      CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS, VKTGS,        APPROCH
                  VTGS)                                                        APPROCH
      VKCAS    = VKAPP                                                         APPROCH
      XLF      = 1.0                                                           APPROCH
   19 HAGL     = HAGL  - DELTAH                                                APPROCH
      FPVTASJ  = FPVTAS                                                        APPROCH
      IF(HAGL.GT.HFLARE) THEN                                                  APPROCH
C****     OBTAIN ATMOSPHERIC VARIABLES.                                        APPROCH
          PRESALT  = HRUNWAY + HAGL                                            APPROCH
          CALL ATMOSPH(PRESALT,ATMARY)                                         APPROCH
          CALL FORCEX(ALPHA,CD,CL)                                             APPROCH
          FPVTAS   = VAPP/SQRT(SIGMA)                                          APPROCH
          FPACCEL  = (FPVTAS - FPVTASJ)/APPDTIM                                APPROCH
          GDIST    = GDIST + DGDIST                                            APPROCH
          HAGLJ    = HAGL                                                      APPROCH
          TIME     = TIME + DBLE(APPDTIM)                                      APPROCH
          XLF      = 1.0                                                       APPROCH
          THETAF   = ALPHA + GAMMAR*RX                                         APPROCH
          CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS, VKTGS,    APPROCH
                      VTGS)                                                    APPROCH
          VKCAS    = VKAPP                                                     APPROCH
          LINENUM  = LINENUM + 1                                               APPROCH
          IF(WRITITR) THEN                                                     APPROCH
              IF(APPDTIM.GE.0.10) THEN                                         APPROCH
                  WRITE(LUOUT,1002) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,     APPROCH
     &                              FPACCEL,CL,CD,THETAF,ALPHA,GAMMAPP,        APPROCH
     &                              DTDT,-SINKRT,XLF,THRUST,XENG               APPROCH
              ELSE                                                             APPROCH
                  WRITE(LUOUT,1003) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,     APPROCH
     &                              FPACCEL,CL,CD,THETAF,ALPHA,GAMMAPP,        APPROCH
     &                              DTDT,-SINKRT,XLF,THRUST,XENG               APPROCH
              ENDIF                                                            APPROCH
          ENDIF                                                                APPROCH
 1002     FORMAT(F6.1,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,          APPROCH
     &            F6.2,F9.0,F7.3)                                              APPROCH
 1003     FORMAT(F6.2,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,          APPROCH
     &            F6.2,F9.0,F7.3)                                              APPROCH
          GO TO 19                                                             APPROCH
      ENDIF                                                                    APPROCH
```

```
C****
C**** CALCULATE TIME PLUS DTIMEJ FOR TIME WITHOUT ALTITUDE
C**** EXTRAPOLATION.  THEN, AFTER GROUND DISTANCE AND TIME ARE
C**** EXTRAPOLATED FOR THE EXACT FLARE HEIGHT, HFLARE, CALCULATE
C**** NEW DELTA TIME (WHICH WILL ALWAYS BE LESS THAN THE ORIGINAL
C**** DELTA TIME, DTIMEJ) SO THAT THE ELAPSED TIME WILL BE AN EVEN
C**** MULTIPLE OF THE ORIGINAL DELTA TIME, DTIMEJ.  LATER IN
C**** SUBROUTINE FLARE, DTIME WILL BE RESET TO THE ORIGINAL DELTA
C**** TIME, DTIMEJ
      TTAPPDT  = FLOAT(TIME) + APPDTIM
      TTDTIME  = FLOAT(TIME + DTIME)
C****
C**** EXTRAPOLATE VALUES FOR DISTANCE AND TIME TO CORRESPOND TO
C**** ALTITUDE HFLARE.
      GDIST = INTERP(GDIST+DGDIST          ,GDIST        ,HAGL,HAGLJ,HFLARE)
      TIME  = INTERP(FLOAT(TIME+APPDTIM) ,FLOAT(TIME),HAGL,HAGLJ,HFLARE)
      IF(APPDTIM.GT.FLOAT(DTIME)) THEN
         IF(TTDTIME-FLOAT(TIME).LE.ZERO) THEN
            DTIME = DBLE(TTAPPDT) - TIME
         ELSE
            DTIME = DBLE(TTDTIME) - TIME
         ENDIF
         IMINUS = INT(APPDTIM/FLOAT(DTIMEJ)) + 1
         DO 10 I=1,IMINUS
            IF(DTIME.GT.DTIMEJ) DTIME = DTIME - DTIMEJ
 10      CONTINUE
      ELSE
         IF(TTAPPDT-FLOAT(TIME).LE.ZERO) THEN
            DTIME = DBLE(TTDTIME) - TIME
         ELSE
            DTIME = DBLE(TTAPPDT) - TIME
         ENDIF
         IMINUS = INT(FLOAT(DTIMEJ)/APPDTIM) + 1
         DO 20 I=1,IMINUS
            IF(FLOAT(DTIME).GT.APPDTIM) DTIME = DTIME - DBLE(APPDTIM)
 20      CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE FLARE(ALPHA,GDIST,ODIST,ROCTD,VKTGS)                   FLARE      1
C**** THIS SUBROUTINE CALCULATES THE FLARE MANEUVER BY RUNGE-KUTTA       FLARE      2
C**** NUMERICAL INTEGRATION.  THE INTEGRATION STEP SIZE IS DTIME         FLARE      3
C**** SECONDS.  THE INPUT VARIABLE, HFLARE, IS THE HEIGHT AT WHICH THE   FLARE      4
C**** FLARE IS STARTED.                                                  FLARE      5
C****                                                                    FLARE      6
C**** THE VARIABLE DTDTGEX IS PROVIDED TO THE USER AS A MEANS TO ACCOUNT FLARE      7
C**** FOR A LOSS IN THE PITCH RATE CAPABILITY DUE TO GROUND EFFECT.      FLARE      8
C**** DTDTGEX MAY BE A FUNCTION OF WING HEIGHT/WING SURFACE AREA, TOTAL  FLARE     19
C**** LIFT COEFFICIENT, CL; CIRCULATION LIFT COEFFICIENT; ETC.  DTDTGEX  FLARE     10
C**** WOULD EQUAL 1.0 OUT OF GROUND EFFECT AND LESS THAN 1.0 IN GROUND   FLARE     11
C**** EFFECT.  DTDTGEX SHOULD BE CALCULATED IN SUBROUTINE "FXXAERO",     FLARE     12
C**** PASSED THROUGH THE CALLING STATEMENT TO SUBROUTINE FORCEX. THE     FLARE     13
C**** USER MAY IGNORE DTDTGEX WITHOUT AFFECTING PROGRAM EXECUTION.       FLARE     14
C****                                                                    FLARE     15
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                  CTRL      1
      PARAMETER (LUIN=3,LUOUT=4)                                         CTRL      2
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL  3
     &                  NEQ,NPAGE,TIME,TIMEROL                           CTRL      4
      COMMON/AIRCRFT/   AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ, AIRCRFT 1
     &                  LOADING,SWING,THTMAX,WNGLOD,XLFMAX               AIRCRFT   2
      COMMON/AERO/      CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,  AERO      1
     &                  SPDBRK,SPOILER,VKCAS,VKTAS                       AERO      2
      COMMON/ENGINE/    AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,   ENGINE    1
     &                  REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,     ENGINE    2
     &                  XENGOUT,XIDLE,XMIL,ZFN                           ENGINE    3
      COMMON/AIRBORN/   ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ AIRBORN  1
      COMMON/AIRSPED/   VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX, AIRSPED  1
     &                  VKMCG,VKROTAT,VKSTART,VKWIND,VWIND               AIRSPED   2
      COMMON/RUNWAY/    ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,   RUNWAY    1
     &                  HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD, RUNWAY 2
     &                  TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU              RUNWAY    3
      DOUBLEPRECISION   FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS, FPINTEG   1
     &                  ROC,RKAIR                                        FPINTEG   2
      COMMON/FPINTEG/   FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS, FPINTEG   3
     &                  ROC,RKAIR(40)                                    FPINTEG   4
      COMMON/ATMOS/     TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF ATMOS 1
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                   CONST     1
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                   CONST     2
      COMMON/CONST/     ASLSQR5,TWOOVR7                                  CONST     3
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,  FLAGS     1
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,    FLAGS     2
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,    FLAGS     3
     &          VFFLAG,WRITITR                                           FLAGS     4
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,  FLAGS     5
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,  FLAGS     6
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS  7
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                   FLAGS     8
```

```
      LOGICAL INIRKI,INIRKJ                                             FLARE
      REAL ATMARY(8),S(48),INTERP                                       FLARE
      EQUIVALENCE (TEMPR,ATMARY(1)),(FPVTAS,S(1))                       FLARE
      EXTERNAL DERIVAL,INTERP                                           FLARE
C****                                                                   FLARE
C**** AIRBORNE INTEGRATION VARIABLES                                     FLARE
C**** NEQ      = NUMBER OF EQUATIONS                                    FLARE
C**** DTIME    = TIME INTERVAL, STEP SIZE (SECONDS)                     FLARE
C**** FPVTAS   = VELOCITY ALONG FLIGHT PATH (FEET/SECOND)               FLARE
C**** GAMMAR   = FLIGHT PATH ANGLE (RADIANS)                            FLARE
C**** FPDIST   = DISTANCE ALONG FLIGHT PATH (FEET)                      FLARE
C**** PRESALT  = ALTITUDE (FEET)                                        FLARE
C**** FPACCEL  = ACCELERATION ALONG FLIGHT PATH (FEET/SECOND**2)        FLARE
C**** DGDTR    = TIME RATE OF CHANGE OF FLIGHT PATH ANGLE               FLARE
C****            (RADIANS/SECOND)                                       FLARE
C**** VHAS     = HORIZONTAL SPEED (FEET/SECOND)                         FLARE
C**** ROC      = RATE OF CLIMB (FEET/SECOND)                            FLARE
C****                                                                   FLARE
C**** INITIALIZE VARIABLES FOR FLARE INTEGRATION LOOP.                  FLARE
      GAMAPPR = GAMMAPP/RX                                              FLARE
      GAMMAR  = GAMAPPR - ASIN((VWIND/FPVTAS)*SIN(GAMAPPR))             FLARE
      HAGL    = HFLARE                                                  FLARE
      ICOUNT  = 1                                                       FLARE
      INIRKI  = .FALSE.                                                 FLARE
      ODIST   = ZERO                                                    FLARE
C****                                                                   FLARE
C**** FLARE INTEGRATION LOOP                                            FLARE
   19 DO 10 NCOUNT=ICOUNT,10                                            FLARE
         PRESALT  = HAGL + HRUNWAY                                      FLARE
         ALPHAJ   = ALPHA                                               FLARE
C****                                                                   FLARE
C****    OBTAIN ATMOSPHERIC VARIABLES.                                  FLARE
         CALL ATMOSPH(PRESALT,ATMARY)                                   FLARE
         CALL FORCEX(ALPHA,CD,CL)                                       FLARE
         CALL SPEED( GAMMAR,FPVTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,VKTGS,  FLARE
     *               VTGS)                                              FLARE
```

```
      IF(.NOT.INIRKI) THEN
C****    MAKE INITIAL INTEGRATION STEP.
         CALL INTX(NEQ,TIME,DTIME,S,DERIVAL,ALPHA)
         ROC    = FPVTAS*SIN(GAMMAR)
         SINKRT = -ROC*60.0
         THETAF = ALPHA + GAMMAR*RX
         GAMMA  = ATAN(ROC/VTGS)*RX
         DTDT   = ZERO
         IF(DTIMEJ.EQ.DTIME) THEN
C****       NO SHORT TIME INCREMENT IN THIS CALL TO FLARE.
            INIRKI = .TRUE.
            INIRKJ = .FALSE.
         ELSE
C****       A SHORT TIME INCREMENT IN THIS CALL TO FLARE IS NEEDED
C****       TO BRING THE ELAPSED TIME TO AN EVEN MULTIPLE OF DELTA
C****       TIME FOR THIS INTEGRATION STEP ONLY.
            INIRKI = .TRUE.
            INIRKJ = .TRUE.
         ENDIF
      ELSE
C****    INCREASE ANGLE OF ATTACK AT A RATE OF DADTCMD*DTDTGEX
C****    DEGREES PER SECOND.
         CALL PITCH('LANDING',ALPHA,DADTCMD,DTIME,DTDTGEX,LUOUT)
C****
C****    MAKE INTEGRATION STEP.
         CALL INTZ(NEQ,TIME,DTIME,S,DERIVAL,ALPHA)
         IF(ERRFLAG) THEN
C****       TERMINATE LANDING EXECUTION.
            TERMFLG = .TRUE.
            RETURN
         ELSEIF(INIRKJ) THEN
C****       A SHORT TIME INCREMENT IN THIS CALL TO FLARE WAS NEEDED
C****       TO BRING THE ELAPSED TIME TO AN EVEN MULTIPLE OF DELTA
C****       TIME FOR THIS INTEGRATION STEP ONLY; RESET DTIME TO THE
C****       ORIGINAL VALUE, DTIMEJ. SET DTIMEJ TO ZERO SO THE
C****       FLAGS INIRKI AND INIRKJ WILL BE RESET TO PROPER VALUES.
            INIRKI  = .FALSE.
            ICOUNT  = ICOUNT - 1
            DTIME   = DTIMEJ
         ENDIF
         GAMMA = ATAN(ROC/VTGS)*RX
         GDISTJ  = GDIST
         CALL INTG( DIST,RKAIR(27),DTIME,GAMMAR,VWIND,0.0,
     *              DDIST,GDIST,GWT)
```

```
            HAGLJ  = HAGL
            HAGL   = PRESALT - HRUNWAY
            SINKRT = -ROC*60.0
            TIMEJ  = FLOAT(TIME - DTIME)
C****
C****       OBTAIN SINK RATE AT TOUCHDOWN.
            IF     (HAGL.LE.0.10) THEN
C****              STORE CURRENT TIME IN TTDTIME VARIABLE. THEN, AFTER
C****              GROUND DISTANCE AND TIME ARE INTERPOLATED FOR THE
C****              TOUCHDOWN POINT, CALCULATE NEW DELTA TIME (WHICH
C****              WILL ALWAYS BE LESS THAN THE ORIGINAL DELTA TIME,
C****              DTIMEJ) SO THAT THE ELAPSED TIME WILL BE AN EVEN
C****              MULTIPLE OF THE ORIGINAL DELTA TIME, DTIMEJ. LATER IN
C****              SUBROUTINE ROLL, DTIME WILL BE RESET TO THE ORIGINAL
C****              DELTA TIME, DTIMEJ.
                   NPTS    = INT(TIME/DTIME)
                   ICOUNT  = NPTS - INT(10.0*FLOAT(NPTS/10)) - 1
                   TTDTIME = FLOAT(TIME)
                   GDIST   = INTERP(GDIST        ,GDISTJ ,HAGL,HAGLJ,ZERO)
                   ROCTD   = INTERP(ROC          ,ROCJ   ,HAGL,HAGLJ,ZERO)
                   TIME    = INTERP(FLOAT(TIME)  ,TIMEJ  ,HAGL,HAGLJ,ZERO)
                   DTIME   = DBLE(TTDTIME) - TIME
                   RETURN
            ELSEIF (ROC.GE.ZERO) THEN
                   NPTS    = INT(TIME/DTIME)
                   ICOUNT  = NPTS - INT(10.0*FLOAT(NPTS/10)) - 1
                   ROCTD   = ZERO
                   RETURN
            ENDIF
            ROCJ = ROC
        ENDIF
```

```
            IF(HAGL.LT.ZERO) HAGL = 0.10
            IF(OVERFLG .AND. HAGL.LT.HCLEAR) THEN
C****           INTERPOLATE TO FIND DISTANCE AT OBSTACLE CLEARANCE HEIGHT,
C****           HCLEAR, IF THE FLARE HEIGHT, HFLARE, IS HIGHER THAN THE
C****           OBSTACLE CLEARANCE HEIGHT, HCLEAR.
                OVERFLG = .FALSE.
                ODIST = INTERP(GDIST,GDISTJ,HAGL,HAGLJ,HCLEAR)
            ENDIF
            IF(WRITITR .AND. (.NOT. INIRKJ .OR. INIRKI)) THEN
                IF(DTIME.GE.0.10D0) THEN
                    WRITE(LUOUT,1002) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,
     &                                FPACCEL,CL,CD,THETAF,ALPHA,GAMMA,
     &                                DTDT,-SINKRT,XLF,THRUST,XENG
                ELSE
                    WRITE(LUOUT,1003) TIME,GDIST,GWT,HAGL,VKCAS,VKTAS,VKTGS,
     &                                FPACCEL,CL,CD,THETAF,ALPHA,GAMMA,
     &                                DTDT,-SINKRT,XLF,THRUST,XENG
                ENDIF
                LINENUM = LINENUM + 1
                IF(LINENUM.GE.NPAGE) THEN
C****               RESET LINE NUMBER COUNTER LINENUM AND WRITE HEADER FOR
C****               NEW PAGE.
                    LINENUM = 3
                    WRITE(LUOUT,1001)
                ENDIF
            ENDIF
1001        FORMAT('1 TIME  DIST.  WEIGHT   ALT.  VCAS  VTAS  VTGS',
     &             ' ACCEL    CL    CD THETA ALPHA GAMMA DTHET ',
     &             '  R/C LOAD  THRUST  XENG'/,
     &             '(SEC) (FEET)  (LBS) (FEET) (KTS) (KTS) (KTS)',
     &             '(FPS2)          (DEG) (DEG) (DEG)  /DT ',
     &             '(FPM) FACT   (LBS) OR MU'/)
1002        FORMAT(F6.1,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,
     &             F6.2,F9.0,F7.3)
1003        FORMAT(F6.2,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,
     &             F6.2,F9.0,F7.3)
  10    CONTINUE
        GO TO 19
        END
```

```
      SUBROUTINE ROLL(ALPHA,GDIST)                                        ROLL
C**** THIS SUBROUTINE CONTROLS THE EXECUTION OF THE GROUND ROLL OF THE   ROLL
C**** LANDING MANEUVER OR THE DECELERATION PORTION OF A REFUSED TAKEOFF  ROLL
C**** MANEUVER.  SUBROUTINE ROLL CALLS INTX TO PERFORM THE NUMERICAL     ROLL
C**** INTEGRATION OF THE RESULTANTS OF THE EQUATIONS OF MOTION AND CALLS ROLL
C**** FORCEX TO OBTAIN THE FORCE COEFFICIENTS FOR THE EQUATIONS           ROLL
C**** OF MOTION.                                                          ROLL
C****                                                                     ROLL
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                   CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                          CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                            CTRL
      COMMON/AIRCRFT/   AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,  AIRCRFT
     &                  LOADING,SWING,THTMAX,WNGLOD,XLFMAX                AIRCRFT
      COMMON/AERO/      CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,   AERO
     &                  SPDBRK,SPOILER,VKCAS,VKTAS                        AERO
      COMMON/ENGINE/    AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,    ENGINE
     &                  REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,      ENGINE
     &                  XENGOUT,XIDLE,XMIL,ZFN                            ENGINE
      COMMON/AIRSPED/   VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX, AIRSPED
     &                  VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                AIRSPED
      COMMON/RUNWAY/    ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,    RUNWAY
     &                  HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,  RUNWAY
     &                  TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU               RUNWAY
      DOUBLEPRECISION   VTAS,DIST,ACCEL,VTASJ,RKGRND                      INTEG
      COMMON/INTEG/     VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                  INTEG
      COMMON/ATMOS/     TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF  ATMOS
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                    CONST
     &            RX=57.29577951308,TSLF=59.0,ZERO=0.0)                   CONST
      COMMON/CONST/     ASLSQR5,TWOOVR7                                   CONST
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,   FLAGS
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,     FLAGS
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,     FLAGS
     &          VFFLAG,WRITITR                                            FLAGS
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,   FLAGS
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,   FLAGS
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,  FLAGS
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                    FLAGS
      CHARACTER  ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,    CHARV
     &           THRCRV*3,TKOTYPE*7                                       CHARV
      COMMON/CHARV/     ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,  CHARV
     &                  TKOTYPE                                           CHARV
```

```
      LOGICAL IDLE,IDLFLAG,INIRKI,INIRKJ,SPOOLF,THROTTL                 ROLL
      REAL ATMARY(8),T(24)                                              ROLL
      EQUIVALENCE (TEMPR,ATMARY(1)),(VTAS,T(1))                         ROLL
      EXTERNAL DERIVGR                                                  ROLL
      DATA  AOAABRK, BRAKMU,IMU, RCR, TIMEBRK,TIMEFLP,TIMEIDL, TIMEREV  ROLL
     &      /     0.0,     0.250,  0,  0.0,     3.0,    999.,     3.0,       0.0 /,  ROLL
     &         TIMESBK, TIMESPL,   VKABRK                               ROLL
     &      /      0.0,        0.0,        0.0 /                        ROLL
      NAMELIST/ROL/BRAKMU,IMU,RCR                                       ROLL
      NAMELIST/ROL2/AOAABRK,BRKFCTR,TIMEBRK,TIMEFLP,TIMEIDL,TIMEREV,TIMESPL,  ROLL
     &                    VKABRK,VKBRAKE                                ROLL
C****                                                                   ROLL
C**** RESET SOME INPUT PARAMETERS FROM ANY PREVIOUS RUN (IF ANY).        ROLL
C**** PARAMETERS NOT RESET HERE RETAIN THEIR VALUE FOR THE SUBSEQUENT    ROLL
C**** RUN IF NOT EXPLICITLY DECLARED IN NAMELIST ROL AND ROL2.           ROLL
      BRKFCTR = 1.0                                                     ROLL
      VKBRAKE = 999.                                                    ROLL
C****                                                                   ROLL
C**** INPUT DATA LOADED INTO ROLL THRU NAMELIST /ROL/ AND /ROL2/.        ROLL
      READ(LUIN,ROL)                                                    ROLL
      READ(LUIN,ROL2)                                                   ROLL
C****                                                                   ROLL
C**** INITIALIZE CURVE FILES.                                            ROLL
      CALL INICURV                                                      ROLL
C****                                                                   ROLL
C**** INITIALIZE LOAD FACTOR TO 1.0.                                     ROLL
      XLF = 1.0                                                         ROLL
      IF(.NOT.RTOFLAG) THEN                                             ROLL
C****       LANDING                                                     ROLL
C****                                                                   ROLL
C****       INITIALIZE RUNGE-KUTTA INTEGRATION FLAG.                     ROLL
            INIRKI      = .FALSE.                                       ROLL
            ICOUNT      = ICOUNT - 1                                    ROLL
C****                                                                   ROLL
C****       INITIALIZE ENGINE THROTTLING FLAGS AND VARIABLES.            ROLL
            IDLFLAG  = .FALSE.                                          ROLL
            REVRSE   = .FALSE.                                          ROLL
            IDLE     = .TRUE.                                           ROLL
            THROTTL  = .TRUE.                                           ROLL
            XENGEND = FLOAT(NENG)*XIDLE                                 ROLL
            XENGTRN = XENG                                              ROLL
```

```
      ELSE
C****       REFUSED TAKEOFF
C****
C****       INITIALIZE RUNGE-KUTTA INTEGRATION FLAG.
            INIRKI = .TRUE
C****
C****       INITIALIZE ENGINE THROTTLING FLAGS AND VARIABLES.
            IF  (NENG.GT.1) THEN
               IDLE   = .FALSE.
            ELSE
               IDLE   = .TRUE.
            ENDIF
            IDLFLAG  = .TRUE.
            REVRSE   = .FALSE.
            SPOOLF   = .FALSE.
            THROTTL  = .TRUE.
C****
C****       FOR THE FAILED ENGINE(S)
            FENGTRN = XENGFLD
            IF      (FAILST.EQ.'IDLE') THEN
                 FENGEND = XIDLE
            ELSEIF(FAILST.EQ.' MIL') THEN
                 FENGEND = XMIL
            ELSEIF(FAILST.EQ.' OFF') THEN
                 FENGEND = ZERO
            ELSE
                 FENGEND = 1.0 -XENGFLD
            ENDIF
            IF  (FAILMOD.EQ.'SEIZE') THEN
               SPOOLF = .TRUE.
               XENG    = XENGOUT
               XENGF   = FENGEND
            ELSE
               XENGF   = 1.0
            ENDIF
C****
C****       FOR THE REMAINING ENGINE(S)
            RENGTRN = FLOAT(INT(XENGOUT))
            RENGEND = FLOAT(NENG - INT(XENGFLD))*XIDLE
            XENGR    = FLOAT(INT(XENGOUT))
            XENGTRN = XENGOUT
            XENG     = XENGF + XENGR
      ENDIF
C****
C**** INITIALIZE AERODYNAMIC VARIABLES.
      CALL ATMOSPH(HRUNWAY,ATMARY)
      CALL FORCEX(ALPHA,CD,CL)
C****
C**** RUNGE-KUTTA INTEGRATION OF GROUND ROLL.
      NEQ  = 2
      HAGL = ZERO
```

```
C****
C**** GROUND ROLL INTEGRATION LOOP
      IF(ICOUNT.EQ.11) ICOUNT = 1
   19 DO 10 NCOUNT=ICOUNT,10
           IF(.NOT.INIRKI) THEN
C****           INITIALIZE INTEGRATION VARIABLES.
                CALL INTX(NEQ,TIME,DTIME,T,DERIVGR,ALPHA)
                CALL INTG( DIST,DISTJ,DTIME,0.0,VWIND,WFUEL,
                           DDIST,GDIST,GWT)
                IF(DTIMEJ.EQ.0.0D0) THEN
C****                NO SHORT TIME INCREMENT IN THIS CALL TO ROLL.
                     INIRKI = .TRUE.
                     INIRKJ= .FALSE.
                ELSE
C****                A SHORT TIME INCREMENT IN THIS CALL TO ROLL IS NEEDED
C****                TO BRING THE ELAPSED TIME TO AN EVEN MULTIPLE OF DELTA
C****                TIME FOR THIS INTEGRATION STEP ONLY.
                     INIRKI = .TRUE.
                     INIRKJ= .TRUE.
                ENDIF
```

```
      ELSE                                                            ROLL
        IF(.NOT.(AOA0FLG)) THEN                                       ROLL
C****     FOR AIRCRAFT NOT AT A THREE POINT ATTITUDE, SET THE         ROLL
C****     TARGET ANGLE OF ATTACK TO EITHER AOAABRK OR AOA3PT          ROLL
C****     DEGREES.                                                    ROLL
          ALPHAJ = ALPHA                                              ROLL
          IF(AOAABRK.EQ.ZERO .OR. VKABRK.EQ.ZERO)  THEN               ROLL
            ALPHTGT = AOA3PT                                          ROLL
          ELSE                                                        ROLL
            IF(VKCAS.GT.VKABRK) THEN                                  ROLL
              ALPHTGT = AOAABRK                                       ROLL
            ELSE                                                      ROLL
              ALPHTGT = AOA3PT                                        ROLL
            ENDIF                                                     ROLL
          ENDIF                                                       ROLL
          DALPHA = ABS(ALPHTGT - ALPHA)                               ROLL
          IF(DALPHA.GT.DADTCMD*FLOAT(DTIME)) THEN                     ROLL
C****       CHANGE ANGLE OF ATTACK UNTIL EQUAL TO ALPHTGT.            ROLL
            IF      (ALPHA.LT.ALPHTGT) THEN                           ROLL
              CALL PITCH('ROLL   ',ALPHA, DADTCMD,DTIME,DTDTGEX,      ROLL
     &                  LUOUT)                                        ROLL
            ELSEIF (ALPHA.GE.ALPHTGT) THEN                            ROLL
              CALL PITCH('ROLL   ',ALPHA,-DADTCMD,DTIME,DTDTGEX,      ROLL
     &                  LUOUT)                                        ROLL
            ENDIF                                                     ROLL
            DADT    = (ALPHA - ALPHAJ)/FLOAT(DTIME)                   ROLL
          ELSE                                                        ROLL
C****       IF THE ANGLE OF ATTACK IS WITHIN ONE PITCH INCREMENT      ROLL
C****       OF THE TARGET ANGLE OF ATTACK, SET ALPHA EQUAL TO         ROLL
C****       ALPHTGT.                                                  ROLL
            ALPHA      = ALPHTGT                                      ROLL
            DADT       = ZERO                                         ROLL
            DADTCMD = ZERO                                            ROLL
            IF(ALPHTGT.EQ.ZERO) AOA0FLG = .TRUE.                      ROLL
          ENDIF                                                       ROLL
          IF(QS*CY.GE.GW[illegible]*COSD(GAMMARW)) THEN               ROLL
C****       LIFTOFF CRITERION MET.                                    ROLL
            LIFTOFF   = .TRUE.                                        ROLL
            LINENUM = LINENUM + 2                                     ROLL
            XMU        = ZERO                                         ROLL
            WRITE(LUOUT,1001) ALPHA,VKCAS,ROLLMU                      ROLL
1001        FORMAT(' *** RUN-TIME ERROR.  LIFT GREATER THAN'          ROLL
     &             ' WEIGHT DURING AEROBRAKING.'/,                    ROLL
     &             '  ALPHA = ',F6.2,'  CAS =',F7.1,' KNOTS ',        ROLL
     &             '  COEFFICIENT OF FRICTION = ',F6.3)               ROLL
          ELSE                                                        ROLL
            LIFTOFF = .FALSE.                                         ROLL
            XMU       = ROLLMU                                        ROLL
          ENDIF                                                       ROLL
        ENDIF                                                         ROLL
```

```
      TIMEROL = FLOAT(TIME) - TIMEFLD
      IF(.NOT.(BRKFLAG) .AND.
     &   (VKCAS.LT.VKBRAKE .AND. TIMEROL.GE.TIMEBRK)) THEN
C****       BRAKE APPLICATION
            BRKFLAG  = .TRUE.
            BRKENGY = 0.5*(GWT/G)*VTAS**2
            XMU = BRAKMU
            LINENUM  = LINENUM + 1
            WRITE(LUOUT,1002) TIME,GDIST,BRAKMU,VKCAS,VKTAS
 1002       FORMAT( ' BRAKE APPLICATION  (TIME = ',F6.2,
     &                ' DIST = ',F8.1,'  BRAKING COEFFICIENT = ',
     &                F6.3,'  CAS = ',F7.1,' KNOTS  TAS = ',
     &                F7.1,' KNOTS)')
      ENDIF
      IF(RTOFLAG) THEN
C****       FAILED ENGINE TRANSIENT
            IF       (DTFAIL.EQ.ZERO .AND. (.NOT.SPOOLF)) THEN
                 CALL SPOOLDNF(TIME,FENGEND,FENGTRN,SPOOLF,XENGF,LUMSG)
C****
C****            SUM ENGINE MULTIPLICATIVE FACTORS.
                 XENG = XENGF + XENGR
            ELSEIF (DTFAIL.NE.ZERO .AND. (.NOT.SPOOLF)) THEN
                 XENGF = FENGTRN - (FENGTRN - FENGEND)*TIMEROL/DTFAIL
C****
C****            SUM ENGINE MULTIPLICATIVE FACTORS.
                 XENG = XENGF + XENGR
                 IF(XENGF.LE.FENGEND) THEN
                    SPOOLF = .TRUE.
                    XENGF   = FENGEND
                 ENDIF
            ENDIF
      ELSE
            XENGF = ZERO
      ENDIF
      IF((NCOUNT.EQ.10.OR.STEADY) .AND.
     &   (JDEBUG.EQ.6666.OR.JDEBUG.EQ.9999)) THEN
            LINENUM = LINENUM + 2
            WRITE(LUMSG,6666)  THROTTL,IDLFLAG,SPOOLF,IDLE,
     &                         XENGF,XENGR,XENG
 6666       FORMAT(' THROTTL,  IDLFLAG,  SPOOLF,  IDLE,  XENGF,XENGR,XENG =',
     &              /,              2L8,        L7,     L5,   3F10.3)
      ENDIF
```

```
            IF(THROTTL) THEN                                                ROLL
                IF   ((IDLFLAG) . AND. TIMEROL .GE.TIMEIDL)    THEN         ROLL
C****               RETARD THROTTLE TO IDLE.                                ROLL
                    IF(XENGFLD.NE.1.0 .AND. (.NOT. SPOOLF)) THEN            ROLL
C****                   FOR PART POWER ENGINE FAILURES MOVE THE REMAINING   ROLL
C****                   XENGF COMPONENT TO XENGR AND RESET XENGF TO ZERO.   ROLL
                        SPOOLF = .TRUE.                                     ROLL
                        XENGR  = XENGF + XENGR                              ROLL
                        XENGF  = ZERO                                       ROLL
                        IF(DTFAIL.NE.0.0) THEN                              ROLL
                           RENGTRN = XENGOUT                                ROLL
                        ELSE                                                ROLL
                           RENGTRN = (RENGEND*TIMEROL - XENGR*DTFAIL)/      ROLL
     &                               (TIMEROL - DTFAIL)                     ROLL
                        ENDIF                                               ROLL
                    ENDIF                                                   ROLL
                    IF      (DTFAIL.EQ.ZERO .AND. (.NOT. IDLE))    THEN     ROLL
                        CALL SPOOLDNR(TIME,RENGEND,RENGTRN,IDLE,XENGR,      ROLL
     &                                LUMSG)                                ROLL
C****                                                                       ROLL
C****                   SUM ENGINE MULTIPLICATIVE FACTORS.                  ROLL
                        XENG = XENGF + XENGR                                ROLL
                    ELSEIF (DTFAIL.NE.ZERO .AND. (.NOT.IDLE))    THEN       ROLL
                        XENGR = RENGTRN - (RENGTRN - RENGEND)*              ROLL
     &                                    TIMEROL/DTFAIL                    ROLL
                        IF(XENGR.LE.RENGEND) THEN                           ROLL
                            IDLE    = .TRUE.                                ROLL
                            XENGR  = RENGEND                                ROLL
                        ENDIF                                               ROLL
C****                                                                       ROLL
C****                   SUM ENGINE MULTIPLICATIVE FACTORS.                  ROLL
                        XENG = XENGF + XENGR                                ROLL
                    ELSE                                                    ROLL
                        IDLFLAG = .FALSE.                                   ROLL
                        ENGGRP = 'AEI'                                      ROLL
                        IF    ((.NOT.REVFLAG) .AND. (.NOT.RTOFLAG))  THEN   ROLL
                            XENG = FLOAT(NENG)                              ROLL
                        ELSEIF(.NOT.REVFLAG)                         THEN   ROLL
                            IF(FAILST.EQ.'IDLE ') THEN                      ROLL
                                XENG = FLOAT(NENG)                          ROLL
                            ELSE                                            ROLL
                                XENG = FLOAT(NENG - INT(XENGFLD))           ROLL
                            ENDIF                                           ROLL
                        ENDIF                                               ROLL
                    ENDIF                                                   ROLL
```

```
               ELSEIF((REVFLAG) . AND. TIMEROL .GE.TIMEREV)THEN
C****               ADVANCE THROTTLE FOR REVERSE THRUST.
                    IF(.NOT. REVRSE) THEN
                         IF(.NOT. RTOFLAG) ENGGRP = 'AER'
                         CALL SPOOLUP(REVNDX,TIME,0.0,XENGTRN,REVRSE,XENGR)
C****
C****                    SUM ENGINE MULTIPLICATIVE FACTORS.
                         XENG = XENGF + XENGR
                    ELSE
                         REVFLAG = .FALSE.
                    ENDIF
               ELSEIF((.NOT.IDLFLAG) .AND. (.NOT.REVFLAG))    THEN
                    THROTTL = .FALSE.
               ENDIF
          ENDIF
C****
C****     OBTAIN FORCE COEFFICIENTS, CX AND CY.
          CALL FORCEX(ALPHA,CD,CL)
C****
C****     MAKE INTEGRATION STEP.
          CALL INTZ(NEQ,TIME,DTIME,T,DERIVGR,ALPHA)
          CALL INTG( DIST,RKGRND(14),DTIME,0.0,VWIND,WFUEL,
                     DDIST,GDIST,GWT)
          CALL SPEED(0.0,VTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,VKTGS,
                       VTGS)
          ACLSUM   = ACLSUM + ACCEL
          BRKENGY = BRKENGY
     &               + (THRUST - CD*QS - GWT*ROLLMU*COS(GRW))*DDIST
          IF(INIRKJ) THEN
C****          A SHORT TIME INCREMENT IN THIS CALL TO ROLL WAS NEEDED
C****          TO BRING THE ELAPSED TIME TO AN EVEN MULTIPLE OF DELTA
C****          TIME FOR THIS INTEGRATION STEP ONLY; RESET DTIME TO THE
C****          ORIGINAL VALUE, DTIMEJ.  SET DTIMEJ TO ZERO SO THE
C****          FLAGS INIRKI AND INIRKJ WILL BE RESET TO PROPER VALUES.
               INIRKI   = .FALSE.
               DTIME    = DTIMEJ
               DTIMEJ  = DBLE(ZERO)
          ENDIF
```

```
               IF   (TIMEROL.GT.90.0)   THEN
C****               LIMIT ON GROUND ROLL OF SIXTY SECONDS.
                    WRITE(LUOUT,1003) BRAKMU,BRKFCTR,IMU
1003                FORMAT(' *** FAILED IN SUBROUTINE ROLL.'/,
  &                       ' BRAKMU =',F6.3,'  BRKFCTR =',F6.3,'  IMU =',
  &                       I3)
                    RETURN
               ELSEIF(VKTGS.LT. 1.0)     THEN
C****               WITH GROUND SPEED LESS THAN 1.0 FPS, ASSUME GROUND ROLL
C****               ENDS AT THE CURRENT DISTANCE BUT AN ADDITIONAL 0.2
C****               SECONDS.  WRITE FINAL OUTPUT.
                    TIME = TIME + 0.2D0
                    IF(DTIME.GE.0.10D0) THEN
                         WRITE(LUOUT,1006)TIME,GDIST,GWT,ZERO,ZERO,ZERO,ZERO,
  &                                       ACCEL,CL,CD,ZERO,ALPHA,ZERO,ZERO,
  &                                       ZERO,XLF,THRUST,XMU
                    ELSE
                         WRITE(LUOUT,1007)TIME,GDIST,GWT,ZERO,ZERO,ZERO,ZERO,
  &                                       ACCEL,CL,CD,ZERO,ALPHA,ZERO,ZERO,
  &                                       ZERO,XLF,THRUST,XMU
                    ENDIF
C****
C****               CALCULATE AVERAGE DECELERATIONS ABAR AND ABARG FOR
C****               RUNGE-KUTTA INTEGRATION OF GROUND ROLL.
                    ABAR   = -ACLSUM/FLOAT(TIME/DTIME)
                    ABARG = ABAR/G
                    WRITE(LUOUT,1004) BRAKMU,BRKFCTR,IMU,BRKENGY
1004                FORMAT(' BRAKMU =',F6.3,'  BRKFCTR =',F6.3,'  IMU =',
  &                       I3,'  BRAKE ENERGY =',1PE12.5E2,' FOOT-POUNDS')
                    RETURN
               ELSEIF(NCOUNT.EQ.10)   THEN
C****               CALCULATE ADDITIONAL OUTPUT PARAMETERS.
                    THETAF = ALPHA
                    IF(LINENUM.GE.NPAGE) THEN
C****                    RESET LINE NUMBER COUNTER LINENUM AND WRITE HEADER
C****                    FOR NEW PAGE.
                         LINENUM = 4
                         WRITE(LUOUT,1005)
                    ENDIF
               ENDIF
          ENDIF
   10 CONTINUE
```

```
C****
C**** WRITE OUTPUT AND INCREMENT LINE NUMBER COUNTER LINENUM.
      IF(DTIME.GE.0.10D0) THEN
         WRITE(LUOUT,1006) TIME,GDIST,GWT,ZERO,VKCAS,VKTAS,VKTGS,ACCEL,
     &                     CL,CD,THETAF,ALPHA,ZERO,DADT,ZERO,XLF,THRUST,
     &                     XMU
      ELSE
         WRITE(LUOUT,1007) TIME,GDIST,GWT,ZERO,VKCAS,VKTAS,VKTGS,ACCEL,
     &                     CL,CD,THETAF,ALPHA,ZERO,DADT,ZERO,XLF,THRUST,
     &                     XMU
      ENDIF
      LINENUM = LINENUM + 1
C**** RESET INTEGRATION LOOP COUNTER ICOUNT TO 1 AND RESTART
C**** THE GROUND ROLL INTEGRATION LOOP.
      ICOUNT = 1
      GO TO 19
1005 FORMAT('1 GROUND ROLL CONTINUED'/,
     &        ' TIME  DIST.  WEIGHT  ALT.  VCAS  VTAS  VTGS',
     &        ' ACCEL   CL   CD THETA ALPHA GAMMA DTHET ',
     &        ' R/C LOAD  THRUST  XENG'/,
     &        '(SEC) (FEET)  (LBS) (FEET) (KTS) (KTS) (KTS)',
     &        '(FPS2)          (DEG) (DEG) (DEG)  /DT ',
     &        '(FPM) FACT  (LBS) OR MU'/)
1006 FORMAT( F6.1,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,F6.2,
     &        F9.0,F7.3)
1007 FORMAT( F6.2,F8.0,F9.0,F8.1,3F7.1,F7.2,F8.4,F7.4,4F7.2,F8.1,F6.2,
     &        F9.0,F7.3)
      END
```

```
      SUBROUTINE INTX(NEQ,TIME,DTIME,T,DERIV,ALPHA)
C**** THIS SUBROUTINE PERFORMS NUMERICAL INTEGRATION USING A FOURTH
C**** ORDER RUNGE-KUTTA SCHEME WITH THE ADAMS-BASHFORTH-MOULTON
C**** PREDICTOR-CORRECTOR METHOD.
C****
C**** REFERENCE: MORRIS, JOHN LL.
C****           COMPUTATIONAL METHODS IN ELEMENTARY NUMERICAL ANALYSIS
C****           COPYRIGHT 1983 BY JOHN WILEY AND SONS LTD.
C****           LIBRARY OF CONGRESS CATALOG QA297.M647 519.4
C****           ISBN 0 471 10419 1
C****           PP 388-97
C****
      DOUBLE PRECISION DTIME,DTIME24,HK,HK2,HK4,RKUTTA(4),T(1),TIME
      INTEGER N(11)
      DO 10 I=2,11
 10      N(I) = NEQ*I
      NCOUNT = N(6)
      DTIME24 = DTIME/24.0D0
      CALL DERIV(ALPHA)
      DO 20 I=1,N(2)
 20      T(I+N(4)) = T(I)
      RETURN
      ENTRY INTZ(NEQ,TIME,DTIME,T,DERIV,ALPHA)
      IF (NCOUNT.LE.N(10)) THEN
         RKUTTA(1) = RKUTTA(2) = TIME + DTIME*0.5D0
         RKUTTA(3) = RKUTTA(4) = TIME + DTIME
         DO 40 J=1,4
            TIME = RKUTTA(J)
            DO 30 I=1,NEQ
               HK2 = DTIME*T(I+NEQ)
               HK  = 0.5D0*HK2
               HK4 = 2.0D0*HK2
               IF     (J.EQ.1) THEN
                  T(I+N(2))  = T(I)
                  T(I+N(3))  = HK2
                  T(I)       = T(I+N(2)) + HK
               ELSEIF (J.EQ.2) THEN
                  T(I)       = T(I+N(2)) + HK
                  T(I+N(3))  = T(I+N(3)) + HK4
               ELSEIF (J.EQ.3) THEN
                  T(I)       = T(I+N(2)) + HK2
                  T(I+N(3))  = T(I+N(3)) + HK4
               ELSEIF (J.EQ.4) THEN
                  T(I)       = T(I+N(2)) + (T(I+N(3)) + HK2)/6.0D0
               ENDIF
 30         CONTINUE
            CALL DERIV(ALPHA)
 40      CONTINUE
         DO 50 I=1,N(2)
 50         T(I+NCOUNT) = T(I)
         NCOUNT = NCOUNT + N(2)
```

```
      ELSE
         DO 60 I=1,NEQ
            T(I+N(3))  = 19.0D0*T(I+N(11)) - 5.0D0*T(I+N(9)) + T(I+N(7))
60          T(I)       = T(I+N(10)) + DTIME24*(55.0D0*T(I+N(11))
     &                   - 59.0D0*T(I+N(9)) + 37.0D0*T(I+N(7))
     &                   -  9.0D0*T(I+N(5)))
         TIME = TIME + DTIME
         CALL DERIV(ALPHA)
         DO 70 I=1,NEQ
            T(I+N(2))  = T(I)
70          T(I)       = T(I+N(10))
     &                   + DTIME24*(9.0D0*T(I+NEQ) + T(I+N(3)))
         CALL DERIV(ALPHA)
         DO 80 I=1,N(6)
80          T(I+N(4)) = T(I+N(6))
         DO 90 I=1,N(2)
90          T(I+N(10)) = T(I)
      ENDIF
      RETURN
      END
```

Subroutine INTG

```
      SUBROUTINE INTG(DIST,DISTJ,DTIME,GAMMAR,VWIND,WFUEL,DDIST,GDIST,GWT)
C**** THIS SUBROUTINE CALCULATES INCREMENTAL GROUND DISTANCE, DDIST;
C**** TOTAL GROUND DISTANCE, GDIST; AND GROSS WEIGHT, GWT, USING FINITE
C**** DIFFERENCE INTEGRATION EQUATIONS.
C****
      DOUBLE PRECISION DTIME
      DDIST = (DIST - DISTJ)*COS(GAMMAR) - VWIND*FLOAT(DTIME)
      GDIST = GDIST + DDIST
      GWT   = GWT - (WFUEL/3600.)*FLOAT(DTIME)
      RETURN
      END
```

```
      SUBROUTINE DERIVGR(ALPHA)                                          DERIVGR
C**** THIS SUBROUTINE DERIVGR CALCULATES THE ACCELERATION, ACCEL, FOR    DERIVGR
C**** THE GROUND ROLL OF A TAKEOFF, LANDING OR REFUSED TAKEOFF.          DERIVGR
C****                                                                    DERIVGR
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                  CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                         CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                           CTRL
      COMMON/AIRCRFT/   AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ, AIRCRFT
     &                  LOADING,SWING,THTMAX,WNGLOD,XLFMAX               AIRCRFT
      COMMON/AERO/      CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,  AERO
     &                  SPDBRK,SPOILER,VKCAS,VKTAS                       AERO
      COMMON/AIRSPED/   VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX, AIRSPED
     &                  VKMCG,VKROTAT,VKSTART,VKWIND,VWIND               AIRSPED
      COMMON/RUNWAY/    ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,   RUNWAY
     &                  HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD, RUNWAY
     &                  TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU              RUNWAY
      DOUBLEPRECISION   VTAS,DIST,ACCEL,VTASJ,RKGRND                     INTEG
      COMMON/INTEG/     VTAS,DIST,ACCEL,VTASJ,RKGRND(20)                 INTEG
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                   CONST
     &             RX=57.29577951308,TSLF=59.0,ZERO=0.0)                 CONST
      COMMON/CONST/     ASLSQR5,TWOOVR7                                  CONST
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,  FLAGS
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,    FLAGS
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,    FLAGS
     &          VFFLAG,WRITITR                                           FLAGS
      COMMON/FLAGS/     AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,  FLAGS
     &                  FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,  FLAGS
     &                  REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                  TERMFLG,VECTFLG,VFFLAG,WRITITR                   FLAGS
      REAL LODMAIN,LODNOSE                                               DERIVGR
      DATA ITER/9/                                                       DERIVGR
C****                                                                    DERIVGR
C**** OBTAIN FORCE COEFFICIENTS CX AND CY.                               DERIVGR
      CALL FORCEX(ALPHA,CD,CL)                                           DERIVGR
      IF(BRKFLAG .AND. IMU.NE.0) THEN                                    DERIVGR
         TIMEB = TIMEFLD + TIMEBRK                                       DERIVGR
         IF(FLOAT(TIME).GE.TIMEB) THEN                                   DERIVGR
            VKTGS = (VTAS - VWIND)/FPSKTS                                DERIVGR
C****                                                                    DERIVGR
C****       CALCULATE WEIGHT MINUS LIFT.                                 DERIVGR
            WTML = GWT*COS(GRW) - CY*QS                                  DERIVGR
            CALL GENMU(VKTGS,WTML,XMAIN,XNOSE,YCG)                       DERIVGR
```

```
               DO 10 I=1,ITER
                  LODMAIN  = ((GWT*(ACCEL/G + SIN(GRW)) + CX*QS)*YCG
     &                         + WTML*XNOSE)/(XMAIN + XNOSE)
                  LODNOSE = WTML - LODMAIN
                  FEX      = -(GWT*SIN(GRW) + CX*QS
     &                         + BRAKMU*LODMAIN - ROLLMU*LODNOSE)
                  ACCEL    = FEX*(G/GWT)
                  IF(ABS(ACCEL-ACCELJ).GE.0.01) THEN
                     ACCELJ = ACCEL
                  ELSE
                     VTASJ   = VTAS
                     RETURN
                  ENDIF
 10            CONTINUE
               WRITE(LUOUT,1001) ACCEL,ACCELJ,LODMAIN,LODNOSE
1001           FORMAT(' *** UNABLE TO CONVERGE ON SOLUTION FOR EQUATION',
    &                 ' OF MOTION WITH NOSE AND MAIN GEAR FORCES',
    &                 ' RESOLVED.  ACCEL,ACCELJ,LODMAIN,LODNOSE =',
    &                 2F10.2,2F10.0,' ***')
               RETURN
            ENDIF
      ENDIF
C****
C**** CALCULATE GROUND ROLL ACCELERATION (FT/SEC**2), ACCEL.
      FEX     = -(GWT*(SIN(GRW) + XMU*COS(GRW)) + QS*(CY*XMU-CX))
      ACCEL  = FEX*(G/GWT)
      VTASJ  = VTAS
      RETURN
      END
```

```
      SUBROUTINE DERIVAT(ALPHA)                                          DERIVAT
C**** THIS SUBROUTINE CALCULATES THE TIME DERIVATIVES FOR THE AIRBORNE   DERIVAT
C**** PORTION OF THE TAKEOFF AND MANAGES THE FLIGHTPATH CONTROL.         DERIVAT
C****                                                                    DERIVAT
      COMMON/AIRCRFT/  AOA2PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ, AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                AIRCRFT
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,   AERO
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                        AERO
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ AIRBORN
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,  FPINTEG
     &                 ROC,RKAIR                                         FPINTEG
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,  FPINTEG
     &                 ROC,RKAIR(40)                                     FPINTEG
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                   CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                   CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                   CONST
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,   FLAGS
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,     FLAGS
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,     FLAGS
     &         VFFLAG,WRITITR                                            FLAGS
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,   FLAGS
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,   FLAGS
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                    FLAGS
      DATA DALPHA/0.05/                                                  DERIVAT
      ALPHMX   = THTMAX                                                  DERIVAT
      ITER     = INT((ALPHMX + 2.0)/DALPHA)                              DERIVAT
      DO 20 I=1,ITER                                                     DERIVAT
C****                                                                    DERIVAT
C****    OBTAIN FORCE COEFFICIENTS CX AND CY.                            DERIVAT
         CALL FORCEX(ALPHA,CD,CL)                                        DERIVAT
C****                                                                    DERIVAT
C****    CALCULATE LOAD FACTOR.                                          DERIVAT
         XLF = (QS*CY)/GWT + (1.0 - COS(GAMMAR))                         DERIVAT
C****                                                                    DERIVAT
C****    CHECK FUSELAGE ANGLE. IF THE FUSELAGE ANGLE IS GREATER THAN     DERIVAT
C****    THE MAXIMUM FUSELAGE ANGLE, REDUCE ANGLE OF ATTACK.             DERIVAT
         THETAF =  ALPHA + GAMMAR*RX                                     DERIVAT
```

```
          IF(THETAF.GT.THTMAX) THEN                                  DERIVAT
              ALPHA = THTMAX - GAMMAR*RX                             DERIVAT
          ELSE                                                       DERIVAT
C****         CHECK FLIGHTPATH ACCELERATION. IF THE ACCELERATION IS  DERIVAT
C****         LESS THAN ZERO, REDUCE ANGLE OF ATTACK.                DERIVAT
              FPACCEL = (G/GWT)*(-CX*QS - GWT*SIN(GAMMAR))           DERIVAT
              IF(FPACCEL.LT.ZERO) THEN                               DERIVAT
                  ALPHA = ALPHA - DALPHA                             DERIVAT
                  IF(CL.LT.0.0000) THEN                              DERIVAT
                      WRITE(LUOUT,1001) FPACCEL,THTMAX               DERIVAT
1001                  FORMAT(' *** UNABLE TO MAINTAIN FLIGHTPATH',   DERIVAT
  &                         ' ACCELERATION GREATER THAN OR EQUAL TO', DERIVAT
  &                         ' ZERO.  FPACCEL,THTMAX =',F10.3,F10.1,' ***') DERIVAT
                      ERRFLAG = .TRUE.                               DERIVAT
                      RETURN                                         DERIVAT
                  ENDIF                                              DERIVAT
              ELSE                                                   DERIVAT
C****             CALCULATE DGAMMA/DT, HORIZONTAL SPEED AND RATE OF  DERIVAT
C****             CLIMB, DGDTR, VHAS AND ROC.                        DERIVAT
                  DGDTR    = (G/(GWT*FPVTAS))*(CY*QS - GWT*COS(GAMMAR)) DERIVAT
                  VHAS     = FPVTAS*COS(GAMMAR)                      DERIVAT
                  ROC      = FPVTAS*SIN(GAMMAR)                      DERIVAT
                  RETURN                                             DERIVAT
              ENDIF                                                  DERIVAT
          ENDIF                                                      DERIVAT
   20 CONTINUE                                                       DERIVAT
      END                                                            DERIVAT
```

```
      SUBROUTINE DERIVAL(ALPHA)                                              DERIVAL
C**** THIS SUBROUTINE CALCULATES THE TIME DERIVATIVES FOR THE AIRBORNE      DERIVAL
C**** PORTION OF THE LANDING AND MANAGES THE FLIGHTPATH CONTROL.            DERIVAL
C****                                                                       DERIVAL
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                      CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                             CTRL
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                 NEQ,NPAGE,TIME,TIMEROL                                CTRL
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,    AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                    AIRCRFT
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,       AERO
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                            AERO
      COMMON/AIRBORN/  ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ     AIRBORN
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,      FPINTEG
     &                 ROC,RKAIR                                             FPINTEG
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,      FPINTEG
     &                 ROC,RKAIR(40)                                         FPINTEG
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                       CONST
     &             RX=57.29577951308,TSLF=59.0,ZERO=0.0)                     CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                       CONST
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,      FLAGS
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,        FLAGS
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,        FLAGS
     &          VFFLAG,WRITITR                                               FLAGS
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,       FLAGS
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,       FLAGS
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,   FLAGS
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                        FLAGS
      DATA DALPHA/0.05/                                                      DERIVAL
      ITER = INT(2.0*ALPHMX/DALPHA)                                          DERIVAL
      DO 30 I=1,ITER                                                         DERIVAL
C****                                                                       DERIVAL
C****    LIMIT ALPHA TO ALPHMX                                              DERIVAL
         IF(ALPHA.GT.ALPHMX) ALPHA = ALPHMX                                  DERIVAL
C****                                                                       DERIVAL
C****    OBTAIN FORCE COEFFICIENTS CX AND CY.                               DERIVAL
         CALL FORCEX(ALPHA,CD,CL)                                            DERIVAL
C****                                                                       DERIVAL
C****    CHECK LOAD FACTOR.  IF LOAD FACTOR IS GREATER THAN THE             DERIVAL
C****    MAXIMUM LOAD FACTOR, REDUCE ANGLE OF ATTACK.                       DERIVAL
         XLF = (QS*CY)/GWT + (1.0 - COS(GAMMAR))                             DERIVAL
         IF(XLF.GT.XLFMAX) THEN                                              DERIVAL
            ALPHA = ALPHA - DALPHA                                           DERIVAL
            IF(ALPHA.LT.-ALPHMX) THEN                                        DERIVAL
               ERRFLAG = .TRUE.                                              DERIVAL
               WRITE(LUOUT,1002)                                             DERIVAL
 1002          FORMAT(' *** ERROR IN DERIVAL.  ALPHA IS LESS THAN',          DERIVAL
     &                ' -ALPHMX AND XLF IS GREATER THAN XLFMAX. ***')        DERIVAL
               RETURN                                                        DERIVAL
            ENDIF                                                            DERIVAL
```

```
         ELSE                                                        DERIVAL
C****          CALCULATE TIME RATE OF CHANGE OF FLIGHTPATH ANGLE DGDTR AND   DERIVAL
C****          THE PITCH RATE DTDT.  IF DTHETA/DT IS GREATER THAN THE        DERIVAL
C****          MAXIMUM, DTDTMX, REDUCE ANGLE OF ATTACK.                      DERIVAL
               DGDTR    = (G/(GWT*FPVTAS))*(CY*QS - GWT*COS(GAMMAR))         DERIVAL
               DADT     = (ALPHA - ALPHAJ)/FLOAT(DTIME)                      DERIVAL
               DTDT     = DGDTR*RX + DADT                                    DERIVAL
               IF(DTDT.GT.DTDTMX) THEN                                       DERIVAL
                  ALPHA = ALPHA - DALPHA                                     DERIVAL
                  IF(ALPHA.LT.-ALPHMX) THEN                                  DERIVAL
                     ERRFLAG = .TRUE.                                        DERIVAL
                     WRITE(LUOUT,1003)                                       DERIVAL
1003                 FORMAT(' *** ERROR IN DERIVAL.  ALPHA IS LESS THAN',    DERIVAL
   &                        ' -ALPHMX AND DTDT IS GREATER THAN DTDTMX.',     DERIVAL
   &                        ' ***')                                          DERIVAL
                     RETURN                                                  DERIVAL
                  ENDIF                                                      DERIVAL
               ELSE                                                          DERIVAL
C****             CALCULATE ACCELERATION ALONG FLIGHTPATH AND VELOCITY       DERIVAL
C****             COMPONENTS.                                                DERIVAL
                  FPACCEL  = (G/GWT)*(-CX*QS - GWT*SIN(GAMMAR))              DERIVAL
                  VHAS     = FPVTAS*COS(GAMMAR)                              DERIVAL
                  ROC      = FPVTAS*SIN(GAMMAR)                              DERIVAL
                  RETURN                                                     DERIVAL
               ENDIF                                                         DERIVAL
            ENDIF                                                            DERIVAL
   30 CONTINUE                                                               DERIVAL
      END                                                                    DERIVAL
```

Subroutine ERROR

```
      SUBROUTINE ERROR(LUOUT,ROCFPM)
C**** THIS SUBROUTINE IS CALLED DURING A TAKEOFF SIMULATION WHEN THE
C**** FLAG ERRFLAG IS SET TO .TRUE.  PROGRAM EXECUTION IS TERMINATED FOR
C**** THE PRESENT SET OF NAMELIST INPUTS AND WILL CONTINUE IF ADDITIONAL
C**** NAMELIST INPUTS ARE TO BE PROCESSED.
C****
      IF(ROCFPM.LE.0.0) THEN
         WRITE(LUOUT,1001)
      ELSE
         WRITE(LUOUT,1002)
      ENDIF
 1001 FORMAT(  /,' CANNOT CLIMB AT USER INPUT CONDITIONS.',
     &         /,' *** ABNORMAL TERMINATION OF TAKOFF ***')
 1002 FORMAT(/,' *** ABNORMAL TERMINATION OF TAKOFF ***')
      RETURN
      END
```

Subroutine HALT

```
      SUBROUTINE HALT(LUIN,LUMSG,LUOUT,TERMMSG)
C**** THIS SUBROUTINE TERMINATES PROGRAM EXECUTION, CLOSES THE INPUT AND
C**** OUTPUT FILES, AND WRITES A TERMINATION MESSAGE, TERMMSG.
C****
      CHARACTER*(*) TERMMSG
      WRITE(LUOUT,1001) TERMMSG
 1001 FORMAT(/,A)
      CLOSE (UNIT=LUIN)
      CLOSE (UNIT=LUMSG)
      CLOSE (UNIT=LUOUT)
      STOP ' TOLAND NORMAL TERMINATION'
      END
```

```
      SUBROUTINE ATMOSPH(PRESALT,ARRAY)
C**** THIS SUBROUTINE CALCULATES PRESSURE, TEMPERATURE, DENSITY, SPEED
C**** OF SOUND, KINEMATIC VISCOSITY, AND PRESSURE, TEMPERATURE AND
C**** DENSITY RATIOS. THE INPUTS ARE PRESSURE ALTITUDE, PRESALT, AND
C**** THE TEMPERATURE INCREMENT FROM STANDARD DAY, DTEMPF.
C****
C**** ARRAY(1) = TEMPERATURE (DEG R)   ARRAY(6) = PRESSURE RATIO
C**** ARRAY(2) = PRESSURE     (PSF)    ARRAY(7) = DENSITY RATIO
C**** ARRAY(3) = DENSITY  (SLUG/FT3)   ARRAY(8) = TEMPERATURE RATIO
C****
C**** ARRAY(4) = SPEED OF SOUND (FEET/SECOND)
C**** ARRAY(5) = KINEMATIC VISCOSITY (FT**2/SEC)
C****
      COMMON/ATMOS/TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF
      REAL ARRAY(8)
      DATA   TSLR, PSLPSF,      RHOSL, ASLFPS, ASLKTS,       CONST1
     &       / 518.67, 2116.22, 2.3769E-03, 1116.47,  661.48,0.270558E-06/
      IF(PRESALT.LT.36089.0) THEN
         FACTOR = 1.0 - 6.8750E-06*PRESALT
         ARRAY(6) = DELTA = FACTOR**5.2559
         ARRAY(8) = THETA = FACTOR + DTEMPF/518.67
      ELSE
         ARRAY(6) = DELTA = 0.22336*EXP((36089.0-PRESALT)/20786.0)
         ARRAY(8) = THETA = 0.75187 + DTEMPF/518.67
      ENDIF
      ARRAY(7) = SIGMA   = DELTA/THETA
      ARRAY(1) = TEMPR   = THETA*TSLR
      ARRAY(2) = PRESS   = DELTA*PSLPSF
      ARRAY(3) = RHO     = SIGMA*RHOSL
      ARRAY(4) = AFPS    = SQRT(THETA)*ASLFPS
      ARRAY(5) = VISCOSK = CONST1*TEMPR**1.5/
     &                     ((PRESS/144.0)*(1.0 + 198.72/TEMPR))
      RETURN
      END
```

Subroutine SPEED

```
      SUBROUTINE SPEED(GAMMAR,VTASX,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,
                       VKTGS,VTGS)
C**** THIS SUBROUTINE IS CALCULATES DYNAMIC PRESSURE, MACH NUMBER, AND
C**** VELOCITIES.
C****
      COMMON/AIRCRFT/   AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,
     &                  LOADING,SWING,THTMAX,WNGLOD,XLFMAX
      COMMON/ATMOS/TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)
      COMMON/CONST/     ASLSQR5,TWOOVR7
      VKTAS  = VTASX/FPSKTS
      VTGS   = VTASX*COS(GAMMAR) - VWIND
      VKTGS  = VTGS/FPSKTS
      IF (VTASX.GT.ZERO) THEN
         AMACH = VTASX/AFPS
         QS    = 0.5*RHO*VTASX**2*SWING
         VKEAS = VKTAS*SQRT(SIGMA)
         VKCAS = ASLSQR5*SQRT((DELTA*
     &           ((1.0 + 0.2*AMACH**2)**3.5 - 1.0) + 1.0)**(TWOOVR7)
     &           - 1.0)
      ELSE
C ****     IF TRUE AIRSPEED IS NEGATIVE, RESET THESE PARAMETERS TO
C****      ZERO; THIS CAN OCCUR IN A SIMULATION WITH TAILWIND.
         AMACH = VKEAS = VKCAS = ZERO
         QS    = 0.1
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ITRLND(ERROR,ERRORJ,DRIVER,FACTOR,TOLRNCE,JFLAG)          ITRLND
C**** THIS SUBROUTINE IS A ZERO FINDING ROUTINE WHICH VARIES THE           ITRLND
C**** INDEPENDENT VARIABLE BASED ON THE SIGN AND SIZE OF THE ERROR.  THE   ITRLND
C**** LOOP CONTAINING THE CALL TO ITRLND IS EXITED SUCCESSFULLY WHEN       ITRLND
C**** JFLAG = 3 OR WHEN THE MAGNITUDE OF THE ERROR IS LESS THAN THE        ITRLND
C**** TOLERANCE VALUE, TOLRNCE.  ERROR AND ERRORJ ARE THE PRESENT AND      ITRLND
C**** PREVIOUS ERRORS, RESPECTIVELY.  JFLAG IS A VARIABLE WHICH            ITRLND
C**** INDICATES THE STATUS OF THE SEARCH.  JFLAG IS SET TO 0 BEFORE        ITRLND
C**** ITRLND IS CALLED.  AFTER THE FIRST CALL TO ITRLND, JFLAG IS SET TO   ITRLND
C**** 1.  JFLAG IS SET TO 2 ONCE THE ZERO IS ISOLATED BETWEEN THE UPPER    ITRLND
C**** AND LOWER BOUNDARIES, BOUNDU AND BOUNDL, RESPECTIVELY.  JFLAG IS     ITRLND
C**** SET TO 3 AND THE ZERO IS CONSIDER FOUND WHEN THE ABSOLUTE VALUE      ITRLND
C**** OF THE DIFFERENCE BETWEEN BOUNDU AND BOUNDL IS LESS THAN TOLRNCE,    ITRLND
C**** THE TOLERANCE PASSED TO SUBROUTINE ITRLND.                           ITRLND
C****                                                                      ITRLND
C****  DURING THE FIRST CALL TO ITRLND DRIVERJ, THE PREVIOUS DRIVER, IS    ITRLND
C**** SET TO DRIVER AND DRIVER IS THEN MULTIPLIED BY FACTOR.  ON           ITRLND
C**** SUBSEQUENT CALLS TO ITRLND THESE TWO DRIVER VALUES ARE SET TO        ITRLND
C**** EITHER UPPER AND LOWER BOUND OR A MULTIPLICATION OR DIVISION BY      ITRLND
C**** FACTOR IS APPLIED TO DRIVER.  ONCE THE LOCATION OF THE ZERO IS       ITRLND
C**** DETERMINED TO BE ON ONE SIDE OF THE DRIVER OR THE OTHER, DRIVER      ITRLND
C**** IS SET TO EITHER THE UPPER OR LOWER BOUND AS THE SEARCH IS           ITRLND
C**** NARROWED.                                                            ITRLND
C****                                                                      ITRLND
      LOGICAL FFLAG,MFLAG,PFLAG                                            ITRLND
      IF(JFLAG.EQ.0) THEN                                                  ITRLND
          FFLAG = MFLAG = PFLAG = .FALSE.                                  ITRLND
          JFLAG = 1                                                        ITRLND
      ENDIF                                                                ITRLND
      IF      (.NOT.PFLAG) THEN                                            ITRLND
          PFLAG = .TRUE.                                                   ITRLND
          DRIVERJ = DRIVER                                                 ITRLND
          DRIVER  = DRIVER*FACTOR                                          ITRLND
          RETURN                                                           ITRLND
      ELSEIF (.NOT.FFLAG) THEN                                             ITRLND
          IF      (ERROR*ERRORJ.LT.0.0) THEN                               ITRLND
            IF(DRIVERJ.LE.DRIVER) THEN                                     ITRLND
                BOUNDL  = DRIVERJ                                          ITRLND
                BOUNDU  = DRIVER                                           ITRLND
            ELSE                                                           ITRLND
                BOUNDU  = DRIVERJ                                          ITRLND
                BOUNDL  = DRIVER                                           ITRLND
                ERRORJ  = ERROR                                            ITRLND
            ENDIF                                                          ITRLND
            JFLAG   = 2                                                    ITRLND
            FFLAG   = .TRUE.                                               ITRLND
            DRIVER  = (BOUNDU + BOUNDL)/2.0                                ITRLND
```

```
      ELSEIF(.NOT.MFLAG .AND.
&           ((ERROR.LT.0.0 .AND. ERROR.GE.ERRORJ) .OR.
&           (ERROR.GE.0.0 .AND. ERROR.LE.ERRORJ))) THEN
            DRIVERJ  = DRIVER
            DRIVER   = DRIVER*FACTOR
      ELSE
            MFLAG    = .TRUE.
            BOUNDU   = DRIVER
            DRIVERJ  = DRIVER
            DRIVER   = BOUNDU/FACTOR
      ENDIF
      RETURN
   ENDIF
   IF(ERROR*ERRORJ.GT.0.0) THEN
      BOUNDL = DRIVER
      ERRORJ = ERROR
   ELSE
      BOUNDU = DRIVER
   ENDIF
   DRIVER = (BOUNDU + BOUNDL)/2.0
   IF(ABS(BOUNDU-BOUNDL) .LT. TOLRNCE) JFLAG = 3
   RETURN
   END
```

Program Functions

Function DGDT

```
      FUNCTION DGDT(ALPHA,FPVTAS,GAMMAR,GWT)
C**** THIS FUNCTION CALCULATES DG/DT AS A FUNCTION OF ALPHA.  IT
C**** REQUIRES VARIABLES CY, FPVTAS, GAMMAPP, GWT, AND QS.
C****
      COMMON/AERO/       CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,
     &                   SPDBRK,SPOILER,VKCAS,VKTAS
      COMMON/AIRSPED/    VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,
     &                   VKMCG,VKROTAT,VKSTART,VKWIND,VWIND
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,
     &              RX=57.29577951308,TSLF=59.0,ZERO=0.0)
      COMMON/CONST/      ASLSQR5,TWOOVR7
      CALL FORCEX(ALPHA,CD,CL)
      DGDT = (G/(GWT*FPVTAS))*(CY*QS - GWT*COS(GAMMAR))
      RETURN
      END
```

Function DVDT

```
      FUNCTION DVDT(ALPHA,FPVTAS,GAMMAR,GWT)
C**** THIS FUNCTION CALCULATES DV/DT AS A FUNCTION OF ALPHA.  IT
C**** REQUIRES VARIABLES CX, FPVTAS, GAMMAPP, GWT, AND QS.
C****
      COMMON/AERO/       CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,
     &                   SPDBRK,SPOILER,VKCAS,VKTAS
      COMMON/AIRSPED/    VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,
     &                   VKMCG,VKROTAT,VKSTART,VKWIND,VWIND
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,
     &              RX=57.29577951308,TSLF=59.0,ZERO=0.0)
      COMMON/CONST/      ASLSQR5,TWOOVR7
      CALL FORCEX(ALPHA,CD,CL)
      DVDH = DVTDH(FPVTAS,' VC')
      DVDT = G/GWT*(-CX*QS - GWT*SIN(GAMMAR)) - DVDH*FPVTAS*SIN(GAMMAR)
      RETURN
      END
```

```
      FUNCTION DVTDH (VTAS,CONSTANT)                                    DVTDH
C**** THIS FUNCTION CALCULATES THE CLIMB SPEED DERIVATIVE, DV/DH FOR    DVTDH
C**** CONSTANT CALIBRATED AIRSPEED, EQUIVALENT AIRSPEED, OR MACH NUMBER. DVTDH
C****                                                                   DVTDH
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                 CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                        CTRL
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                NEQ,NPAGE,TIME,TIMEROL                            CTRL
      DOUBLEPRECISION FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,  FPINTEG
     &                ROC,RKAIR                                         FPINTEG
      COMMON/FPINTEG/ FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,  FPINTEG
     &                ROC,RKAIR(40)                                     FPINTEG
      COMMON/ATMOS/   TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF ATMOS
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G=32.174,                   CONST
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                  CONST
      COMMON/CONST/   ASLSQR5,TWOOVR7                                   CONST
      COMMON/RACURV/  IDBUG,IEXTOR,KURNAM(99)                           RACURV
      CHARACTER CONSTANT*4                                              DVTDH
      IF(IDBUG.GE.1) WRITE(LUMSG,9001)                                  DVTDH
9001  FORMAT (' ENTERED DVTDH')                                         DVTDH
      CALL SPEED( GAMMAR,VTAS,VWIND,AMACH,QS,VKCAS,VKEAS,VKTAS,         DVTDH
                 VKTGS,VTGS)                                            DVTDH
      IF   (CONSTANT.EQ.' VC') THEN                                     DVTDH
C****    CONSTANT CALIBRATED AIRSPEED                                   DVTDH
         DERIVA = DADH(PRESALT)                                         DVTDH
         DERIVD = DDELTDH(PRESALT)                                      DVTDH
         IF(IDBUG.GE.1) WRITE(LUMSG,1001) DERIVA,DERIVD,TIME            DVTDH
1001     FORMAT(' ***** FROM FUNCTION DVTDH,  DADH,DDELTDH =',2E16.8,   DVTDH
     &             ' AT TIME ',E14.8,' *****')                          DVTDH
         IF        (VKCAS.LE.ASL .AND. AMACH.LE.1.0) THEN               DVTDH
C****         SUBSONIC EQUATION                                         DVTDH
              CONSTX  = 1.0 + 0.2*AMACH**2                              DVTDH
              DVTDH   = AMACH*DERIVA - (5.0*AFPS*((CONSTX)**3.5 - 1.0)* DVTDH
     &                  DERIVD)/(7.0*AMACH*(CONSTX)**2.5)               DVTDH
         ELSEIF    (VKCAS.GT.ASL .AND. AMACH.GT.1.0) THEN               DVTDH
C****         SUPERSONIC EQUATION                                       DVTDH
              CONSTY  = 7.0*AMACH**2 - 1.0                              DVTDH
              DVTDH   = AMACH*DERIVA - (AFPS*DERIVD*(166.921*AMACH**7 - DVTDH
     &                    (CONSTY)**2.5)*(CONSTY))/                     DVTDH
     &                    (1166.45*AMACH**6*(2.0*AMACH**2 - 1.0))       DVTDH
         ELSE                                                           DVTDH
              WRITE(LUMSG,1002)                                         DVTDH
1002          FORMAT(' ILLEGAL VALUES FOR VKCAS AND AMACH PASSED TO',   DVTDH
     &               ' FUNCTION DVTDH.  EITHER VKCAS IS GREATER THAN',/, DVTDH
     &               ' ASL AND AMACH IS LESS THAN 1.0 OR VKCAS IS LESS', DVTDH
     &               ' THAN OR EQUAL TO ASL AND AMACH IS GREATER THAN',/, DVTDH
     &               ' 1.0; EXECUTION RETURNS TO CALLING SUBROUTINE.')  DVTDH
         ENDIF                                                          DVTDH
```

```
      ELSEIF(CONSTANT.EQ.' VE') THEN                                  DVTDH
C****    CONSTANT EQUIVALENT AIRSPEED                                 DVTDH
         DXDH = DSIGDH(PRESALT)                                       DVTDH
         IF(IDBUG.GE.1) WRITE(LUMSG,1003) DXDH,TIME                   DVTDH
1003     FORMAT(' ***** FROM FUNCTION DVTDH,  DSIGDH =',E16.8,        DVTDH
  &                    ' AT TIME ',E14.8, ' ******')                  DVTDH
         DVTDH = -(VKEAS*FPSKTS)*DXDH/(2.0*SIGMA**1.5)                DVTDH
      ELSEIF(CONSTANT.EQ.'MACH') THEN                                 DVTDH
C****    CONSTANT MACH NUMBER                                         DVTDH
         DXDH = DADH(PRESALT)                                         DVTDH
         IF(IDBUG.GE.1) WRITE(LUMSG,1004) DXDH,TIME                   DVTDH
1004     FORMAT(' ***** FROM FUNCTION DVTDH,  DADH =',E16.8,          DVTDH
  &                    ' AT TIME ',E14.8, ' ******')                  DVTDH
         DVTDH = AMACH*DXDH                                           DVTDH
      ELSE                                                            DVTDH
         WRITE(LUMSG,1005)                                            DVTDH
1005     FORMAT(' ILLEGAL OPTION CODE PASSED TO FUNCTION DVTDH;',     DVTDH
  &                   ' EXECUTION RETURNS TO CALLING SUBROUTINE.')    DVTDH
      ENDIF                                                           DVTDH
      RETURN                                                          DVTDH
      END                                                             DVTDH
```

Function DADH

```
      FUNCTION DADH(HC)
C**** THIS FUNCTION CALCULATES THE SPEED OF SOUND DERIVATIVE WITH
C**** RESPECT TO PRESSURE ALTITUDE.
C****
      DOUBLEPRECISION DTIME,DTIMEJ,TIME
      PARAMETER (LUIN=3,LUOUT=4)
      COMMON/CTRL/          DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,
     &                      NEQ,NPAGE,TIME,TIMEROL
      IF       (HC.GE. -1000.00 .AND. HC.LE. 36089.24) THEN
         DADH = -0.0651520/SQRT(288.15 - .0019812*HC)
      ELSEIF (HC.GT. 36089.24 .AND. HC.LE. 65616.80) THEN
         DADH = 0.0
      ELSEIF (HC.GT. 65616.80 .AND. HC.LE. 104986.88) THEN
         DADH = 0.0100234/SQRT(216.65 + .0003048*(HC - 65616.8))
      ELSEIF (HC.GT. 104986.88 .AND. HC.LE. 154199.48) THEN
         DADH = 0.0280641/SQRT(228.65 + .0008534*(HC - 104986.88))
      ELSE
         WRITE(LUMSG,1001)
1001     FORMAT(' ILLEGAL PRESSURE ALTITUDE PASSED TO FUNCTION DADH;',
     &          ' EXECUTION RETURNED TO CALLING SUBROUTINE')
      ENDIF
      RETURN
      END
```

Function DDELTDH

```
      FUNCTION DDELTDH(HC)
C**** THIS FUNCTION CALCULATES THE PRESSURE RATIO DERIVATIVE WITH
C**** RESPECT TO PRESSURE ALTITUDE.
C****
      DOUBLEPRECISION DTIME,DTIMEJ,TIME
      PARAMETER (LUIN=3,LUOUT=4)
      COMMON/CTRL/        DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,
     &                    NEQ,NPAGE,TIME,TIMEROL
      IF     (HC.GE. -1000.00 .AND. HC.LE. 36089.24) THEN
         DDELTDH = -0.361374E-04/(1.0 - HC*6.875586E-06)
      ELSEIF (HC.GT. 36089.24 .AND. HC.LE. 65616.80) THEN
         DDELTDH = -0.480637E-04
      ELSEIF (HC.GT. 65616.80 .AND. HC.LE. 104986.88) THEN
         DDELTDH = -0.480637E-04/(1.0 + (HC - 65616.80)*1.40688E-06)
      ELSEIF (HC.GT. 104986.88 .AND. HC.LE. 154199.48) THEN
         DDELTDH = -0.455412E-04/(1.0 + (HC - 104986.88)*3.73252E-06)
      ELSE
         WRITE(LUMSG,1001)
1001     FORMAT(' ILLEGAL PRESSURE ALTITUDE PASSED TO FUNCTION',
     &          ' DDELTDH; EXECUTION RETURNED TO CALLING SUBROUTINE')
      ENDIF
      RETURN
      END
```

```
      FUNCTION DSIGDH(HC)
C**** THIS SUBROUTINE CALCULATES THE DENSITY RATIO DERIVATIVE WITH
C**** RESPECT TO PRESSURE ALTITUDE.
C****
      DOUBLEPRECISION DTIME,DTIMEJ,TIME
      PARAMETER (LUIN=3,LUOUT=4)
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,
     &                  NEQ,NPAGE,TIME,TIMEROL
      IF     (HC.GE. -1000.00 .AND. HC.LE. 36089.24) THEN
         DSIGDH = -2.926178E-05*(1.0 - HC*6.87558E-06)**3.2559
      ELSEIF (HC.GT. 36089.24 .AND. HC.LE. 65616.80) THEN
         DSIGDH = EXP (-4.80637E-05*(HC - 36089.24))*(-1.427853E-05)
      ELSEIF (HC.GT. 65616.80 .AND. HC.LE. 104986.88) THEN
         DSIGDH = -3.555171E-06*
     &            (1.0 + 1.40688E-06*(HC - 65616.80))**(-36.1634)
      ELSEIF (HC.GT. 104986.88 .AND. HC.LE. 154199.48) THEN
         DSIGDH = -5.319445E-07*
     &            (1.0 + 3.73252E-06*(HC - 104986.88))**(-14.2012)
      ELSE
         WRITE(LUMSG,1001)
1001     FORMAT(' ILLEGAL PRESSURE ALTITUDE PASSED TO FUNCTION',
     &          ' DSIGDH; EXECUTION RETURNED TO CALLING SUBROUTINE')
      ENDIF
      RETURN
      END
```

Function INTERP

```
      REAL FUNCTION INTERP(YNOW,YPAST,XNOW,XPAST,XANS)
C**** THIS FUNCTION RETURNS THE INTERPOLATED VALUE BETWEEN TWO ORDINATE
C**** INPUTS BASED ON THE ADDITIONAL INPUTS OF THREE ABSCISSAS.
C****
      XDIFF = XNOW - XPAST
C****
C**** CHECK DIFFERENCE BETWEEN X-COORDINATES TO PREVENT ARITHMETIC
C**** DIVISION BY ZERO.
      IF(XDIFF.EQ.0.0) XDIFF = 1.0
      INTERP = YNOW - ((YNOW - YPAST)*(XNOW - XANS)/XDIFF)
      RETURN
      END
```

```
      FUNCTION ZEROX(A,B,FUNCTN,VAR2,VAR3,VAR4,TOLRNCE,FINDV)
C**** THIS FUNCTION FINDS THE ZERO OF THE FUNCTION FUNCTN(X) ON AN
C**** INTERVAL X = A TO X = B.  THE PROGRAM ASSUMES THAT F(AX) AND
C**** F(BX) ARE OF OPPOSITE SIGN, IMPLYING THAT FUNCTN(X) IS ZERO
C**** SOMEWHERE IN THAT INTERVAL.  THE ARGUMENT, "FUNCTN", MUST BE
C**** DECLARED EXTERNAL IN THE MAIN CALLING PROGRAM AND HAVE FOUR
C**** CALLING ARGUMENTS, A OR B, VAR2, VAR3, AND VAR4.
C****
      LOGICAL FINDV
      REAL A,B,FUNCTN,TOLRNCE
      DATA EPSILON/2.22E-16/
      AX   = CX =  A
      BX   =       B
      IF(FINDV) THEN
         FAX  = FCX = FUNCTN(VAR2,AX,VAR3,VAR4)
         FBX  =       FUNCTN(VAR2,BX,VAR3,VAR4)
      ELSE
         FAX  = FCX = FUNCTN(AX,VAR2,VAR3,VAR4)
         FBX  =       FUNCTN(BX,VAR2,VAR3,VAR4)
      ENDIF
      D    = E = BX - AX
   19 IF(ABS(FCX).LT.ABS(FBX)) THEN
         AX   = BX
         BX   = CX
         CX   = AX
         FAX  = FBX
         FBX  = FCX
         FCX  = FAX
      ENDIF
      TOLX     = 2.0*EPSILON*ABS(BX) + TOLRNCE
      DIFCXBX = CX - BX
      HALFDIF = 0.5*DIFCXBX
      IF    ((ABS(HALFDIF).LE.TOLX) .OR. (FBX.EQ.0.0))       THEN
         ZEROX = BX
         RETURN
      ELSEIF((ABS(E).LT.TOLX) .OR. (ABS(FAX).LE.ABS(FBX)))  THEN
         D = E = HALFDIF
      ELSE
         S = FBX/FAX
         IF(AX.EQ.CX) THEN
            P = DIFCXBX*S
            Q = 1.0 - S
         ELSE
            Q = FAX/FCX
            R = FBX/FCX
            P = S*(DIFCXBX*Q*(Q - R) - (BX - AX)*(R - 1.0))
            Q = (Q - 1.0)*(R - 1.0)*(S - 1.0)
         ENDIF
```

```
      IF(P.LE.0.0) THEN
         P = -P
      ELSE
         Q = -Q
      ENDIF
      S = E
      E = D
      IF(( (2.0*P)  .GE.(3.0*HALFDIF*Q - ABS(TOLX*Q))).OR.
&        (    P   .GE.                 ABS(0.5*S*Q))) THEN
         D = E = HALFDIF
      ELSE
         D = P/Q
      ENDIF
   ENDIF
   AX   = BX
   FAX  = FBX
   IF      (ABS(D).GT.TOLX)  THEN
      BX = BX + D
   ELSEIF (CX.GT.BX)         THEN
      BX = BX + TOLX
   ELSE
      BX = BX - TOLX
   ENDIF
   IF(FINDV) THEN
      FBX = FUNCTN(VAR2,BX,VAR3,VAR4)
   ELSE
      FBX = FUNCTN(BX,VAR2,VAR3,VAR4)
   ENDIF
   IF(FBX*FCX/ABS(FCX).GT.0.0) THEN
      CX   = AX
      FCX  = FAX
      D    = E = BX - AX
   ENDIF
   GO TO 19
   END
```

Subroutine INICURV

```
      SUBROUTINE INICURV                                                INICURV   1
C**** THIS SUBROUTINE INITIALIZES THE NAMES, TABLES AND VALUE NAMES OF  INICURV   2
C**** THE RANDOM ACCESS CURVE FILES AND OPENS THE APPROPRIATE CURVES AS INICURV   3
C**** NEEDED.                                                           INICURV   4
C****                                                                  INICURV   5
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                  CONST     1
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                  CONST     2
      COMMON/CONST/   ASLSQR5,TWOOVR7                                   CONST     3
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,  FLAGS     1
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,    FLAGS     2
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,    FLAGS     3
     &         VFFLAG,WRITITR                                           FLAGS     4
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,   FLAGS     5
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,   FLAGS     6
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS   7
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                    FLAGS     8
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,   CHARV     1
     &          THRCRV*3,TKOTYPE*7                                      CHARV     2
      COMMON/CHARV/   ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,  CHARV     3
     &                TKOTYPE                                           CHARV     4
      COMMON/CINDEX/MSTRNDX(3),NDXLEN,LUCURV                            CINDEX    1
      COMMON/RACURV/  IDBUG,IEXTOR,KURNAM(99)                           RACURV    1
      COMMON/VALUES/VALIN(15),VFUEL(4),VIDLE(4),VLIFT(2),VAERO(15)      VALUES    2
      COMMON/TABLES/TFUEL (172),TIDLE (119),TAERO07(72),TAERO08(88),    TABLES    1
     &              TAERO09(79),TAERO10(62),TAERO11(91),TAERO12(76),    TABLES    2
     &              TAERO13(64),TAERO14(73),TAERO15(74),TAERO16(66),    TABLES    3
     &              TAERO17(66),TAERO18(77),TAERO19(73),TAERO20(76),    TABLES    4
     &              TAERO21(76),TAERO22(81),TAERO23(74)                 TABLES    5
      LOGICAL CRVFLAG                                                   INICURV  12
C     INTEGER                                                           INICURV  13
C     REAL                                                              INICURV  14
      DATA  IDBUG, IEXTOR,LUCURV,NDXLEN,NDXTYPE                         INICURV  15
     &      /  0,      0,      2,     3,      0/                        INICURV  16
C****                                                                  INICURV  17
C**** OPEN RANDOM CURVE FILE WITH EXTENDED CORE STORAGE.                INICURV  18
      IF(.NOT.CRVFLAG) THEN                                             INICURV  19
C        CALL OPENMS (LUCURV,MSTRNDX,NDXLEN,NDXTYPE)                    INICURV  20
      ENDIF                                                             INICURV  21
C****                                                                  INICURV  22
C**** INITIALIZE CURVE ARRAYS.                                          INICURV  23
      IF(.NOT.CRVFLAG) THEN                                             INICURV  24
         CRVFLAG = .TRUE.                                               INICURV  25
         NDXTYPE = 0                                                    INICURV  26
C        CALL CURVSET(KURNAM,TABLE,NDIM( X),NAMEIN,NUNITI,NIPIN,        INICURV  27
C    &                NAMEOUT,NUNITO,NDPOUT(1),IDBUG,LUMSG)             INICURV  28
      ENDIF                                                             INICURV  29
      RETURN                                                            INICURV  30
      END                                                               INICURV  31
```

```
      SUBROUTINE FORCEX(ALPHA,CD,CL)                                    FORCEX   1
C**** THIS SUBROUTINE PROVIDES THE TOTAL FORCE COEFFICIENTS ALONG        FORCEX   2
C**** AND NORMAL TO THE FLIGHTPATH BY CALLING USER PROVIDED SUBROUTINE   FORCEX   3
C**** 'FXXAERO' TO DETERMINE LIFT AND DRAG COEFFICIENTS AND USER         FORCEX   4
C**** PROVIDED SUBROUTINES 'FXXENG', SPOOLUP, SPOOLDNF, AND SPOOLDNR TO  FORCEX   5
C**** PROVIDE THRUST AND FUEL FLOW.                                      FORCEX   6
C****                                                                   FORCEX   7
C**** SUBROUTINE FORCEX SHOULD REQUIRE ONLY THE FOLLOWING MODIFICATIONS: FORCEX   8
C****                                                                   FORCEX   9
C****    1) INITIALIZATION OF THE DATA STATEMENT PROVIDING VALUES FOR    FORCEX  10
C****       AIT, AR, B, CGPCT, CLALPH, CONFIG, DTDTMX, FLAP, FLPPCT, FLPPCT, FORCEX  11
C****       HZ, LOADING, NENG, PWRCODE, RC, SWING, THRCRV, THTMAX,       FORCEX  12
C****       XIDLE, XLFMAX, AND XMIL.                                     FORCEX  13
C****                                                                   FORCEX  14
C****    2) CHANGING THE CALLING STATEMENT FOR SUBROUTINES 'FXXAERO'     FORCEX  15
C****       AND 'FXXENG' TO THE USER PROVIDED NAMES WITH THE             FORCEX  16
C****       APPROPRIATE CALLING ARGUMENTS..                              FORCEX  17
C****                                                                   FORCEX  18
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                 CTRL     1
      PARAMETER (LUIN=3,LUOUT=4)                                        CTRL     2
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL     3
     &                NEQ,NPAGE,TIME,TIMEROL                            CTRL     4
      COMMON/AIRCRFT/ AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ, AIRCRFT  1
     &                LOADING,SWING,THTMAX,WNGLOD,XLFMAX                AIRCRFT  2
      COMMON/AERO/    CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS,   AERO     1
     &                SPDBRK,SPOILER,VKCAS,VKTAS                        AERO     2
      COMMON/ENGINE/  AIT,AMACH,DTFAIL,EPR,FE,FG,FGPCT,NENG,PWRCODE,    ENGINE   1
     &                REVNDX,RC,THRUST,VTANGLE,WFUEL,XENG,XENGFLD,      ENGINE   2
     &                XENGOUT,XIDLE,XMIL,ZFN                            ENGINE   3
      COMMON/AIRBORN/ ALPHAJ,ALPHMX,DTDT,GAMMAPP,ROCFPM,THETAF,XLF,XLFJ AIRBORN  1
      COMMON/AIRSPED/ VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX, AIRSPED  1
     &                VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                AIRSPED  2
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG, FLAGS    1
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,   FLAGS    2
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,   FLAGS    3
     &          VFFLAG,WRITITR                                          FLAGS    4
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,   FLAGS    5
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,   FLAGS    6
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY, FLAGS    7
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                    FLAGS    8
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,   CHARV    1
     &          THRCRV*3,TKOTYPE*7                                      CHARV    2
      COMMON/CHARV/   ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,  CHARV    3
     &                TKOTYPE                                           CHARV    4
      DATA  AIT,   AR,    B, CGPCT,  CLALPH, CONFIG, DTDTMX, FLAP, FLPPCT  FORCEX  27
     &    / 0.0, 7.000, 75.0,  25.0,  0.120,    1.0,    7.0, -1.0,  50.0 /, FORCEX  28
     &      HZ, LOADING, NENG, PWRCODE, RC, SWING, THRCRV, THTMAX       FORCEX  29
     &    / 5.50,     1,    4,   4444., 0.0, 800.,  'RC',    15.0 /,    FORCEX  30
     &      XIDLE, XLFMAX, XMIL                                         FORCEX  31
     &    /  0.06,    2.0, 0.50 /                                       FORCEX  32
```

```
      CALL FXXAERO(ALPHA,AMACH,CL,CD,FLAP,SPOILER,VKCAS,XENGOUT)          FORCEX
      IF(.NOT. STEADY) THEN                                               FORCEX
           CALL FXXENG(AIT,AMACH,NENG,PWRCODE,QS,VKCAS,XENG,XIDLE,        FORCEX
                         FE,FG,THRUST,WFUEL,EPR)                          FORCEX
C****                                                                     FORCEX
C****      ADJUST THRUST BY FGPCT FACTOR                                  FORCEX
           FACTOR    = 1.00 + FGPCT/100.                                  FORCEX
           FG        = FG*FACTOR                                          FORCEX
           THRUST    = THRUST*FACTOR                                      FORCEX
           WFUEL     = WFUEL*FACTOR                                       FORCEX
      ENDIF                                                               FORCEX
C****                                                                     FORCEX
C**** CALCULATE NET THRUST COEFFICIENT.                                    FORCEX
      CTX = (FG*COSD (ALPHA+AIT) - FE)/QS                                 FORCEX
      CTY = (FG*SIND  (ALPHA+AIT)  )/QS                                   FORCEX
C****                                                                     FORCEX
C**** ADD USER INCREMENTS TO CL AND CD.                                    FORCEX
      CL = CL + DCLX                                                      FORCEX
      CD = CD + DCDX                                                      FORCEX
C****                                                                     FORCEX
C**** CALCULATE FORCE COEFFICIENTS CX AND CY.                              FORCEX
      CX = CD - CTX                                                       FORCEX
      CY = CL + CTY                                                       FORCEX
      RETURN                                                              FORCEX
      END                                                                 FORCEX
```

Subroutine FXXAERO

```
      SUBROUTINE FXXAERO(ALPHA,AMACH,CL,CD,FLAP,SPOILER,VKCAS,XENGOUT)            FXXAERO    1
C**** THIS SUBROUTINE CALCULATES OR PERFORMS A TABLE LOOKUP TO                    FXXAERO    2
C**** DETERMINE THE LIFT AND DRAG COEFFICIENTS FOR THE AIRCRAFT. THIS              FXXAERO    3
C**** SUBROUTINE AND ITS NAME ARE PROVIDED BY THE USER.                            FXXAERO    4
C****                                                                              FXXAERO    5
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                            CTRL       1
      PARAMETER (LUIN=3,LUOUT=4)                                                   CTRL       2
      COMMON/CTRL/     DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,       CTRL       3
     &                 NEQ,NPAGE,TIME,TIMEROL                                      CTRL       4
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ,          AIRCRFT    1
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX                          AIRCRFT    2
      COMMON/VECTOR/   HVECT,VVECT                                                 VECTOR     1
      COMMON/AIRSPED/  VKABRK,VKAPP,VKBRAKE,VKEND,VKFAIL,VKFLAP,VKFLPMX,           AIRSPED    1
     &                 VKMCG,VKROTAT,VKSTART,VKWIND,VWIND                          AIRSPED    2
      COMMON/RUNWAY/   ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,              RUNWAY     1
     &                 HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,         RUNWAY     2
     &                 TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                         RUNWAY     3
      DOUBLEPRECISION  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,            FPINTEG    1
     &                 ROC,RKAIR                                                   FPINTEG    2
      COMMON/FPINTEG/  FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,            FPINTEG    3
     &                 ROC,RKAIR(40)                                               FPINTEG    4
      COMMON/ATMOS/    TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF       ATMOS      1
      LOGICAL  AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,             FLAGS      1
     &         GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,               FLAGS      2
     &         RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,               FLAGS      3
     &         VFFLAG,WRITITR                                                      FLAGS      4
      COMMON/FLAGS/    AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,             FLAGS      5
     &                 FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,             FLAGS      6
     &                 REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,         FLAGS      7
     &                 TERMFLG,VECTFLG,VFFLAG,WRITITR                              FLAGS      8
      CHARACTER  ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,             CHARV      1
     &           THRCRV*3,TKOTYPE*7                                                CHARV      2
      COMMON/CHARV/    ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,            CHARV      3
     &                 TKOTYPE                                                     CHARV      4
      COMMON/RACURV/   IDBUG,IEXTOR,KURNAM(99)                                     RACURV     1
      COMMON/VALUES/VALIN(15),VFUEL(4),VIDLE(4),VLIFT(2),VAERO(15)                 VALUES     2
      COMMON/TABLES/TFUEL (172),TIDLE (119),TAERO07(72),TAERO08(88),              TABLES     1
     &              TAERO09(79),TAERO10(62),TAERO11(91),TAERO12(76),               TABLES     2
     &              TAERO13(64),TAERO14(73),TAERO15(74),TAERO16(66),               TABLES     3
     &              TAERO17(66),TAERO18(77),TAERO19(73),TAERO20(76),               TABLES     4
     &              TAERO21(76),TAERO22(81),TAERO23(74)                            TABLES     5
      DATA DCDFLP, DCDCG,DCDWML,DCDEOT, DFLAPDT,DSPLRDT,DTGEAR, VKFLPMX             FXXAERO   17
     &    / 0.0000, 0.0000, 0.0000, 0.0000,   3.0,    9.0,    6.0,   225./          FXXAERO   18
C****                                                                              FXXAERO   19
C**** EMPIRICAL FORMULA FOR LANDING GEAR DRAG.                                     FXXAERO   20
C**** IF(DCDLGR.EQ.0.0) DCDLGR = (0.0032/SWING)*GWT**0.80                          FXXAERO   21
```

```
C****                                                                    FXXAERO
C**** INITIALIZE VALIN ARRAY FOR CURVE FILE LOOKUPS.                     FXXAERO
      VALIN(IA) = PRESALT                                                FXXAERO
      VALIN(IB) = AMACH                                                  FXXAERO
      VALIN(IC) = FLTNDX                                                 FXXAERO
      VALIN(ID) = FLAP                                                   FXXAERO
      VALIN(IE) = VKCAS                                                  FXXAERO
      VALIN(IF) = CGPCT                                                  FXXAERO
      VALIN(IG) = ALPHA                                                  FXXAERO
C****                                                                    FXXAERO
C**** DETERMINE LIFT COEFFICIENT, CL, USING CURVE LIFT.                  FXXAERO
C     CALL CURVEL(KURNAM(XX),TABLE,TABLE,VALIN,VLIFT,                    FXXAERO
C  &                 IDBUG,IEXTOR,LUMSG)                                 FXXAERO
      CLSLOPE  = VLIFT(1)                                                FXXAERO
      CLINTCP  = VLIFT(2)                                                FXXAERO
      VALIN(9)  = CL = CLSLOPE*ALPHA + CLINTCP                           FXXAERO
      IF((NCOUNT.EQ.1OOR.STEADY) .AND.                                   FXXAERO
   &    (JDEBUG.EQ.7001.OR.JDEBUG.EQ.7777.OR.JDEBUG.EQ.9999)) THEN       FXXAERO
          WRITE(LUMSG,7001) VLIFT,VALIN(9),VALIN(7),VALIN(8)             FXXAERO
7001      FORMAT(' CLSLOPE,CLINTCP,CL,FLTNDX,FLAP =',3F10.6,2F8.1)       FXXAERO
      ENDIF                                                              FXXAERO
C****                                                                    FXXAERO
C**** FLAP RETRACTION                                                    FXXAERO
      IF((FLAPFLG .AND. (VKCAS.GE.VKFLAP .OR. VKCAS.GE.VKFLPMX)) .OR.    FXXAERO
   &   (FLAPFLG .AND. TIMEROL.GE.(TDELAY+TFLP)              ))           FXXAERO
   &   CALL FRETRAC(FLAP,VKFLAP,DFLAPDT)                                 FXXAERO
C****                                                                    FXXAERO
C**** SPOILER DEPLOYMENT                                                 FXXAERO
      IF(.NOT.(SPLFLAG) .AND. TIMEROL.GE.(TDELAY+TSPL)) THEN             FXXAERO
          SPLREND = 90.0                                                 FXXAERO
          CALL SPOIL('DEPLOY',SPOILER,SPLREND,DSPLRDT)                   FXXAERO
      ENDIF                                                              FXXAERO
      IF(LGRFLAG) THEN                                                   FXXAERO
C****     DETERMINE DELTA DRAG COEFFICIENT FOR LANDING GEAR, DCDLGR,     FXXAERO
C****     USING CURVE LGRDRAG.                                           FXXAERO
C         CALL CURVEL(KURNAM(XX),TABLE,TABLE,VALIN,VAERO(Y),             FXXAERO
C  &                     IDBUG,IEXTOR,LUMSG)                             FXXAERO
          DCDLGR = VAERO(Y)                                              FXXAERO
          IF(HAGL.GE.HGEAR) CALL GRETRAC(DCDLGR,DTGEAR,HAGL)             FXXAERO
      ELSE                                                               FXXAERO
          DCDLGR = 0.0000                                                FXXAERO
      ENDIF                                                              FXXAERO
```

```
C****                                                                FXXAERO
C**** SUM INCREMENTS FROM CURVE FILES                                FXXAERO
      CL   = CLALPH*ALPHA                                            FXXAERO
      CL   = CL + DCLSPL + DCLX                                      FXXAERO
      CD   = CD + DCDSPL + DCDFLP + DCDREN + DCDLGR + DCDGE*(GEFACTR) FXXAERO
     &          + DCDCG + DCDWML + DCDEOT + DCDX                     FXXAERO
      IF   ((NCOUNT.EQ.10.OR.STEADY) .AND.                           FXXAERO
     &      (JDEBUG.EQ.7002.OR.JDEBUG.EQ.7777.OR.JDEBUG.EQ.9999)) THEN FXXAERO
           WRITE(LUMSG,7002) (VAERO(I),I=1,10)                       FXXAERO
7002       FORMAT(' VAERO(1),SPLDEF,DCDSPL,DCDREN,DCDLGR,DCDGE,DCDCG,', FXXAERO
     & '       DCDWML,DCDEOT,DCLSPL = '/,11F10.6)                    FXXAERO
      ENDIF                                                          FXXAERO
      RETURN                                                         FXXAERO
      END                                                            FXXAERO
```

```
      SUBROUTINE GEFFECT(HAGL,SRATIO)                                GEFFECT
C     SUBROUTINE GEFFECT(ALPHA,CL,HAGL,DCLGE,DCDGE)      PARKS       GEFFECT
C**** THIS SUBROUTINE SUPPLIES THE GROUND EFFECT INCREMENTS TO THE    GEFFECT
C**** AERODYNAMIC COEFFICIENTS.  THIS GENERIC SUBROUTINE PROVIDES     GEFFECT
C**** PREDICTED GROUND EFFECT EQUATIONS WHICH CAN BE MODIFIED BY THE  GEFFECT
C**** USER FOR A PARTICULAR AIRCRAFT.                                 GEFFECT
C****                                                                 GEFFECT
      COMMON/AIRCRFT/  AOA3PT,AR,B,CGPCT,CLALPH,CONFIG,DTDTMX,FLT,GWT,HZ, AIRCRFT
     &                 LOADING,SWING,THTMAX,WNGLOD,XLFMAX             AIRCRFT
      COMMON/AERO/     CX,CY,DADTCMD,DCDX,DCLX,DTDTGEX,FLAP,FLPPCT,QS, AERO
     &                 SPDBRK,SPOILER,VKCAS,VKTAS                     AERO
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                CONST
     &            RX=57.29577951308,TSLF=59.0,ZERO=0.0)               CONST
      COMMON/CONST/    ASLSQR5,TWOOVR7                                CONST
      CHARACTER  ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6, CHARV
     &            THRCRV*3,TKOTYPE*7                                  CHARV
      COMMON/CHARV/    ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV, CHARV
     &                 TKOTYPE                                        CHARV
C**** B     = WINGSPAN        HZ      = HEIGHT WING ABOVE TIRES      GEFFECT
C**** AR    = ASPECT RATIO    CLALPH  = DCL/DALPHA (DIMENSIONLESS)   GEFFECT
C**** HAGL  = ABSOLUTE ALTITUDE OF AIRCRAFT TIRES ABOVE RUNWAY       GEFFECT
      PARAMETER (PI=3.141592653589793)                               GEFFECT
C**** HZB = HZ/B                                                      GEFFECT
C**** HB  = (HZ + HAGL)/B                                             GEFFECT
C****                                                                 GEFFECT
C**** SIGMA INTERPOLATION FORMULA: NACA TN D-970                      GEFFECT
C**** "EFFECT OF GROUND PROXIMITY ON THE AERODYNAMIC"-- ETC           GEFFECT
C**** GIVEN BOTH IN-GROUND EFFECT AND OUT OF GROUND EFFECT AERODYNAMICS, GEFFECT
C**** THIS FORMULA PROVIDES A RATIO BETWEEN THEM AS A FUNCTION OF     GEFFECT
C**** ALTITUDE (AGL).  THE COMMENTED EQUATIONS BELOW SHOW HOW THIS RATIO GEFFECT
C**** CAN BE USED IN THE CALLING SUBROUTINE.                          GEFFECT
C**** CDOUT = CD OUT OF GROUND EFFECT;      CD0 = CD AT HAGL = 0.0    GEFFECT
C**** CLOUT = CL OUT OF GROUND EFFECT;      CL0 = CL AT HAGL = 0.0    GEFFECT
      SIGMAZ= (1.0 - 1.32*HZB)/(1.05 + 7.4*HZB)                       GEFFECT
      SIGMA  = (1.0 - 1.32*HB) /(1.05 + 7.4*HB)                       GEFFECT
      SRATIO = AMAX1(SIGMA/SIGMAZ,0.0)                                GEFFECT
C**** CL    = CLOGE + SRATIO*(CL0 - CLOGE)                            GEFFECT
C**** CD    = CDOGE + SRATIO*(CD0 - CDOGE)                            GEFFECT
C**** DCLGE = SRATIO*(CL0 - CLOGE)                                    GEFFECT
C**** DCDGE = SRATIO*(CD0 - CDOGE)                                    GEFFECT
C****                                                                 GEFFECT
C**** PREDICTED GROUND EFFECT BY DR E K PARKS                         GEFFECT
C**** WITHOUT IN-GROUND EFFECT AERODYNAMICS, THIS EQUATION PROVIDES AN GEFFECT
C**** ESTIMATE OF THE GROUND EFFECT INCREMENTS TO THE OUT OF GROUND   GEFFECT
C**** EFFECT AERODYNAMICS.                                            GEFFECT
C     X      = 25.94*HB**2 + 1.0                                      GEFFECT
C     DCDGE = -8.0*CL**2/(AR*PI**3*X**2)                              GEFFECT
C     DCLGE =  4.0*CL*CLALPH/(AR*PI**3*HB)*(X - SQRT(X)*CL))          GEFFECT
      RETURN                                                          GEFFECT
      END                                                             GEFFECT
```

```
      SUBROUTINE FXXENG( AIT,ALPHA,AMACH,NENG,PWRCODE,QS,VKCAS,XENG,        FXXENG    1
     &                   XIDLE,FE,FG,FN,WF,EPR)                            FXXENG    2
C**** THIS SUBROUTINE CALCULATE OR PERFORMS TABLE LOOKUPS TO DETERMINE      FXXENG    3
C**** THE NET THRUST AND FUEL FLOW FOR THE AIRCRAFT.  THIS SUBROUTINE       FXXENG    4
C**** AND ITS NAME ARE PROVIDED BY THE USER.                                FXXENG    5
C****                                                                       FXXENG    6
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                      CTRL     1
      PARAMETER (LUIN=3,LUOUT=4)                                             CTRL     2
      COMMON/CTRL/    DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT,  CTRL     3
     &                NEQ,NPAGE,TIME,TIMEROL                                 CTRL     4
      COMMON/VECTOR/  HVECT,VVECT                                           VECTOR    1
      COMMON/RUNWAY/  ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,        RUNWAY    1
     &                HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD,   RUNWAY    2
     &                TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                   RUNWAY    3
      DOUBLEPRECISION FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,      FPINTEG   1
     &                ROC,RKAIR                                             FPINTEG   2
      COMMON/FPINTEG/ FPVTAS,GAMMAR,FPDIST,PRESALT,FPACCEL,DGDTR,VHAS,      FPINTEG   3
     &                ROC,RKAIR(40)                                         FPINTEG   4
      COMMON/ATMOS/   TEMPR,PRESS,RHO,AFPS,VISCOSK,DELTA,SIGMA,THETA,DTEMPF  ATMOS    1
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                       CONST    1
     &           RX=57.29577951308,TSLF=59.0,ZERO=0.0)                       CONST    2
      COMMON/CONST/   ASLSQR5,TWOOVR7                                        CONST    3
      LOGICAL   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,FPCTFLG,      FLAGS    1
     &          GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,REVRSE,ROTATE,        FLAGS    2
     &          RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,TERMFLG,VECTFLG,        FLAGS    3
     &          VFFLAG,WRITITR                                               FLAGS    4
      COMMON/FLAGS/   AOA0FLG,BRKFLAG,CLRHGT,ERRFLAG,FAILFLG,FLAPFLG,        FLAGS    5
     &                FPCTFLG,GEFLAG,LGRFLAG,LIFTOFF,OVERFLG,REVFLAG,        FLAGS    6
     &                REVRSE,ROTATE,RTOFLAG,SBKFLAG,SPLFLAG,SPOOL,STEADY,    FLAGS    7
     &                TERMFLG,VECTFLG,VFFLAG,WRITITR                         FLAGS    8
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,        CHARV    1
     &          THRCRV*3,TKOTYPE*7                                           CHARV    2
      COMMON/CHARV/   ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,       CHARV    3
     &                TKOTYPE                                                CHARV    4
      COMMON/RACURV/  IDBUG,IEXTOR,KURNAM(99)                               RACURV    1
      COMMON/VALUES/VALIN(15),VFUEL(4),VIDLE(4),VLIFT(2),VAERO(15)          VALUES    2
      COMMON/TABLES/TFUEL (172),TIDLE (119),TAERO07(72),TAERO08(88),        TABLES    1
     &              TAERO09(79),TAERO10(62),TAERO11(91),TAERO12(76),        TABLES    2
     &              TAERO13(64),TAERO14(73),TAERO15(74),TAERO16(66),        TABLES    3
     &              TAERO17(66),TAERO18(77),TAERO19(73),TAERO20(76),        TABLES    4
     &              TAERO21(76),TAERO22(81),TAERO23(74)                     TABLES    5
      LOGICAL SPOOL                                                         FXXENG   18
      REAL ATMARY(8)                                                        FXXENG   19
      EQUIVALENCE (TEMPR,ATMARY(1))                                         FXXENG   20
      DATA DVECTDT/10.0/                                                    FXXENG   21
      FE =      0.                                                          FXXENG   22
      FG = 20000.                                                           FXXENG   23
      CALL ATMOSPH(PRESALT,ATMARY)                                          FXXENG   24
```

```
      IF ((NCOUNT.EQ.10.OR.STEADY) .AND.                                FXXENG
     &   (JDEBUG.EQ.8001.OR.JDEBUG.EQ.8888.OR.JDEBUG.EQ.9999)) THEN     FXXENG
          WRITE(LUMSG,8001) (VALIN(I),I=1,6),SIGMA                      FXXENG
8001      FORMAT(' TAMBF,PRESALT,VTT,DTEMPF,AMACH,PWSET,SIGMA = '/,     FXXENG
     &    F10.1,2F10.0,F10.1,F10.3,F10.1,F10.3)                         FXXENG
      ENDIF                                                             FXXENG
      VALIN(1) = AMACH                                                  FXXENG
      VALIN(2) = EPR                                                    FXXENG
      VALIN(3) = PWRCODE                                                FXXENG
C     CALL CURVEL(KURNAM(XX),TABLE,TABLE,VALIN,VFUEL,                   FXXENG
C                    IDBUG,IEXTOR,LUMSG)                                FXXENG
C****                                                                   FXXENG
C**** VECTORED THRUST ANGLE REDUCTION                                   FXXENG
      IF  (VECTFLG .AND. (HAGL.GE.HVECT .AND. VKCAS.GE.VVECT))          FXXENG
     &    CALL TVECTOR(VTANGLE,HVECT,VVECT,DVECTDT)                     FXXENG
      IF  ((NCOUNT.EQ.10.OR.STEADY) .AND.                               FXXENG
     &     (JDEBUG.EQ.8002.OR.JDEBUG.EQ.8888.OR.JDEBUG.EQ.9999)) THEN   FXXENG
         WRITE(LUMSG,8002) VFUEL                                        FXXENG
8002     FORMAT(' FG,FE,WF,EPR = ',3F10.0,F10.3)                        FXXENG
      ENDIF                                                             FXXENG
      CDRAM = FE/QS                                                     FXXENG
      FN = FG*COSD(ALPHA + AIT) - FE                                    FXXENG
      WF = 4000.                                                        FXXENG
C****                                                                   FXXENG
C**** CALL SPOOLUP FOR ROLLING TAKEOFFS                                 FXXENG
      IF     (TKOTYPE.EQ.'ROLLING' .AND. (.NOT.SPOOL)) THEN             FXXENG
         CALL SPOOLUP(0.0,TIME,FLOAT(NENG)*XIDLE,XENG,SPOOL,XENG)       FXXENG
      ELSEIF(TKOTYPE.EQ.'ROLLING' .AND. (        SPOOL)) THEN           FXXENG
         SPOOL    = .FALSE.                                             FXXENG
         TKOTYPE  = 'SPOOLED'                                           FXXENG
      ENDIF                                                             FXXENG
      RETURN                                                            FXXENG
      END                                                               FXXENG
```

```
      SUBROUTINE SPOOLUP(ENGNDX,TIME,XENG0,XENGTRN,SPOOL,XENG)
C**** THIS SUBROUTINE DETERMINES THE RATIO OF THRUST TO TAKEOFF RATED
C**** THRUST DURING A ROLLING TAKEOFF USING SUBROUTINE TABINT AND DATA
C**** ARRAY SPOOLA . THIS SUBROUTINE IS JUST AN EXAMPLE AND DOES NOT
C**** APPLY TO ALL AIRCRAFT.
C****
      DOUBLE PRECISION TIME
      LOGICAL NDXSET,SPOOL
      REAL SPOOLA(26)
      DATA SPOOLA/  0.0, 2.0, 4.0, 6.0, 8.0,10.0,20.0,30.0,0.0,1.0,
     &               0.37,0.37,0.37,0.37,0.37,0.37,1.00,1.00,
     &               0.37,0.37,0.37,0.37,0.37,0.37,1.00,1.00/
      IF(.NOT.(NDXSET)) THEN
C****     INITIALIZE PREVIOUS SPOOL FACTOR, SPOOLXJ, START TIME, TIMEZ,
C****     AND ENGINE LOOKUP INDEX, ENGNDX.
          NDXSET   =.TRUE.
          SPOOL    =.FALSE.
          SPOOLXJ  =99.0
          TIMEZ    =FLOAT(TIME)
      ENDIF
      TIMEDIF = FLOAT(TIME) - TIMEZ
      IF    (TIMEDIF.GE.0.0 .AND. TIMEDIF.LT.30.0)       THEN
          SPOOLXJ  =SPOOLX
          CALL TABINT(TIMEDIF,SPOOLX,ENGNDX,8,2,SPOOLA(1),INDEX)
          XENG = XENG0 + XENGTRN*SPOOLX
      ELSEIF(TIMEDIF.GT.0.0 .AND. SPOOLXJ.EQ.SPOOLX)  THEN
C****     THROTTLE TRANSIENT COMPLETE.  RESET NDXSET FLAG AND SET STATUS
C****     FLAG TO .TRUE.
          NDXSET  =.FALSE.
          SPOOL   =.TRUE.
      ENDIF
      RETURN
      END
```

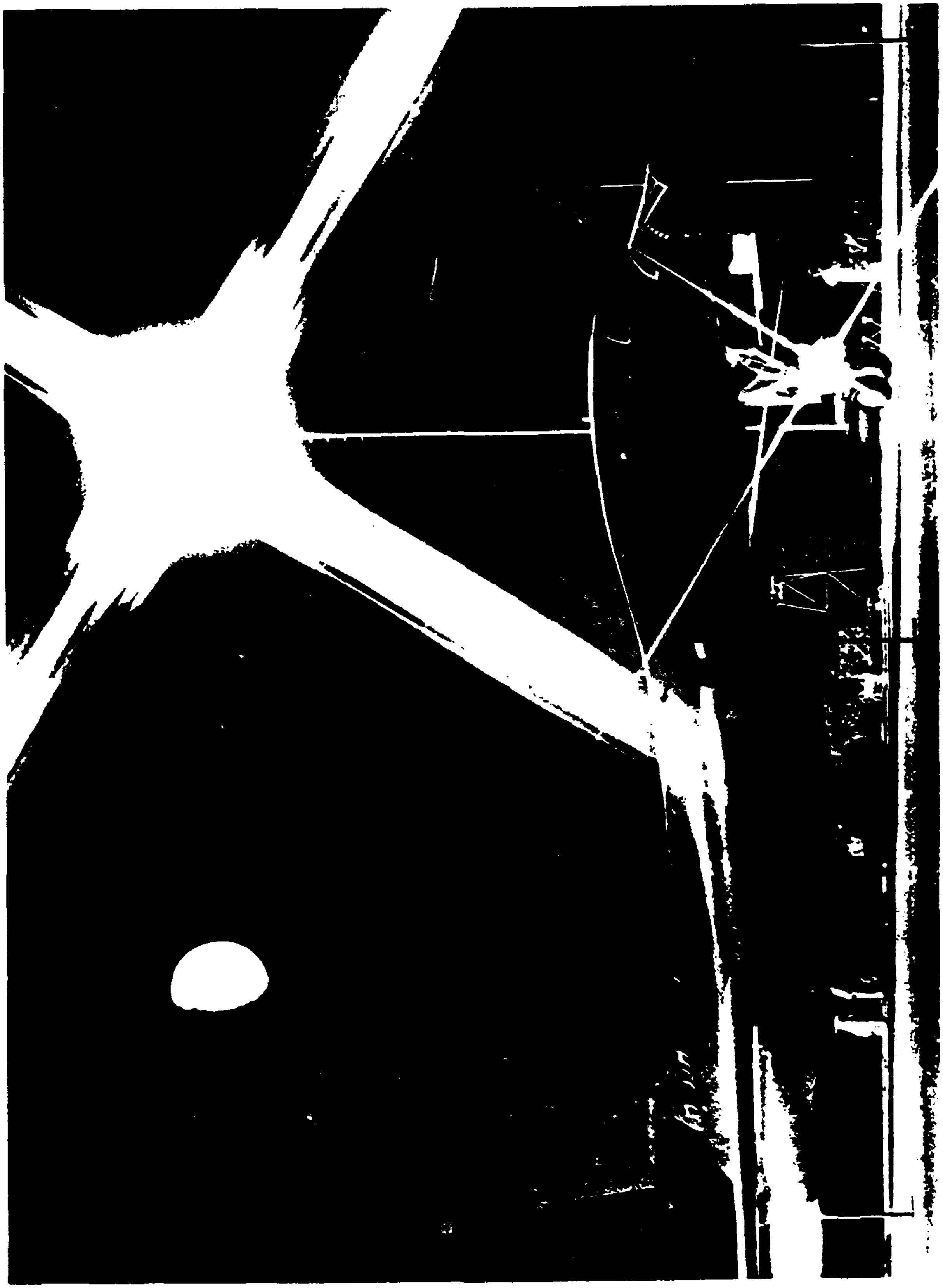
```
      SUBROUTINE SPOOLDNF(TIME,XENGEND,XENGTRN,SPOOL,XENG,LUMSG)
C**** THIS SUBROUTINE DETERMINES THE TOTAL NUMBER OF ENGINES REMAINING
C**** DURING A THROTTLE CHOP OR FUEL CUT USING SUBROUTINE TABINT AND
C**** DATA ARRAY SPOOLA . THIS SUBROUTINE IS JUST AN EXAMPLE AND DOES
C**** NOT APPLY TO ALL AIRCRAFT.
C****
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,
     &             THRCRV*3,TKOTYPE*7
      COMMON/CHARV/  ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,
     &               TKOTYPE
      DOUBLE PRECISION TIME
      LOGICAL NDXSET,SPOOL
      REAL SPOOLA(56)
      DATA SPOOLA/  0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8,
     &              2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.0,10.0,0.00,1.00,
     &             1.00,0.85,0.64,0.44,0.35,0.27,0.23,0.20,0.18,0.17,
     &             0.16,0.14,0.11,0.10,0.08,0.07,0.06,0.06,
     &             1.00,0.85,0.64,0.44,0.31,0.21,0.15,0.12,0.10,0.08,
     &             0.07,0.06,0.04,0.03,0.02,0.01,0.00,0.00/
      IF(.NOT.(NDXSET)) THEN
C****     INITIALIZE PREVIOUS SPOOL FACTOR, SPOOLXJ, START TIME, TIMEZ,
C****     AND ENGINE LOOKUP INDEX, ENGNDX.
          NDXSET  = .TRUE.
          SPOOL   = .FALSE.
          IF      (FAILST.EQ.'IDLE') THEN
              ENGNDX = 0.0
          ELSEIF  (FAILST.EQ.'OFF ') THEN
              ENGNDX = 1.0
          ELSE
              WRITE(LUMSG,1001) FAILST
1001          FORMAT(' INVALID FAILST PASSED TO SPOOLDNF, FAILST = ',A5)
          ENDIF
          SPOOLXJ  = 99.0
          TIMEZ    = FLOAT(TIME)
      ENDIF
      TIMEDIF = FLOAT(TIME) - TIMEZ
      IF(SPOOLXJ.NE.SPOOLX .AND. XENG.GT.XENGEND) THEN
          SPOOLXJ  = SPOOLX
          CALL TABINT(TIMEDIF,SPOOLX,ENGNDX,18,2,SPOOLA(1),INDEX)
          XENG = XENGTRN*SPOOLX
      ELSE
C****     THROTTLE TRANSIENT COMPLETE.  RESET NDXSET FLAG AND SET STATUS
C****     FLAG TO .TRUE.
          NDXSET  = .FALSE.
          SPOOL   = .TRUE.
          XENG    = XENGEND
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SPOOLDNR(TIME,XENGEND,XENGTRN,SPOOL,XENG,LUMSG)       SPOOLDNR   1
C**** THIS SUBROUTINE DETERMINES THE TOTAL NUMBER OF ENGINES REMAINING SPOOLDNR   2
C**** DURING A THROTTLE CHOP OR FUEL CUT USING SUBROUTINE TABINT AND   SPOOLDNR   3
C**** DATA ARRAY SPOOLA. THIS SUBROUTINE IS JUST AN EXAMPLE AND DOES   SPOOLDNR   4
C**** NOT APPLY TO ALL AIRCRAFT.                                       SPOOLDNR   5
C****                                                                  SPOOLDNR   6
      CHARACTER ENGGRP*3,FAILGRP*3,FAILMOD*5,FAILST*4,MVR*3,MANUVR*6,     CHARV   1
     &          THRCRV*3,TKOTYPE*7                                        CHARV   2
      COMMON/CHARV/ ENGGRP,FAILGRP,FAILMOD,FAILST,MVR,MANUVR,THRCRV,      CHARV   3
     &              TKOTYPE                                               CHARV   4
      DOUBLE PRECISION TIME                                            SPOOLDNR   8
      LOGICAL NDXSET,SPOOL                                             SPOOLDNR   9
      REAL SPOOLA(56)                                                  SPOOLDNR  10
      DATA SPOOLA/ 0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8,   SPOOLDNR  11
     &             2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.0,10.0,0.00,1.00,   SPOOLDNR  12
     &            1.00,0.85,0.64,0.44,0.35,0.27,0.23,0.20,0.18,0.17,   SPOOLDNR  13
     &            0.16,0.14,0.11,0.10,0.08,0.07,0.06,0.06,             SPOOLDNR  14
     &            1.00,0.85,0.64,0.44,0.31,0.21,0.15,0.12,0.10,0.08,   SPOOLDNR  15
     &            0.07,0.06,0.04,0.03,0.02,0.01,0.00,0.00/             SPOOLDNR  16
      IF(.NOT.(NDXSET)) THEN                                           SPOOLDNR  17
C****    INITIALIZE PREVIOUS SPOOL FACTOR, SPOOLXJ, START TIME, TIMEZ, SPOOLDNR  18
C****    AND ENGINE LOOKUP INDEX, ENGNDX.                              SPOOLDNR  19
         NDXSET  = .TRUE.                                              SPOOLDNR  20
         SPOOL   = .FALSE.                                             SPOOLDNR  21
         IF     (FAILST.EQ.'IDLE') THEN                                SPOOLDNR  22
            ENGNDX = 0.0                                               SPOOLDNR  23
         ELSEIF (FAILST.EQ.'OFF ') THEN                                SPOOLDNR  24
            ENGNDX = 1.0                                               SPOOLDNR  25
         ELSE                                                          SPOOLDNR  26
            WRITE(LUMSG,1001) FAILST                                   SPOOLDNR  27
1001        FORMAT(' INVALID FAILST PASSED TO SPOOLDNF, FAILST = ',A5) SPOOLDNR  28
         ENDIF                                                         SPOOLDNR  29
         SPOOLXJ = 99.0                                                SPOOLDNR  30
         TIMEZ   = FLOAT(TIME)                                         SPOOLDNR  31
      ENDIF                                                            SPOOLDNR  32
      TIMEDIF = FLOAT(TIME) - TIMEZ                                    SPOOLDNR  33
      IF(SPOOLXJ.NE.SPOOLX .AND. XENG.GT.XENGEND) THEN                 SPOOLDNR  34
         SPOOLXJ = SPOOLX                                              SPOOLDNR  35
         CALL TABINT(TIMEDIF,SPOOLX,ENGNDX,18,2,SPOOLA(1),INDEX)       SPOOLDNR  36
         XENG = XENGTRN*SPOOLX                                         SPOOLDNR  37
      ELSE                                                             SPOOLDNR  38
C****    THROTTLE TRANSIENT COMPLETE. RESET NDXSET FLAG AND SET STATUS SPOOLDNR  39
C****    FLAG TO .TRUE.                                                SPOOLDNR  40
         NDXSET  = .FALSE.                                             SPOOLDNR  41
         SPOOL   = .TRUE.                                              SPOOLDNR  42
         XENG    = XENGEND                                             SPOOLDNR  43
      ENDIF                                                            SPOOLDNR  44
      RETURN                                                           SPOOLDNR  45
      END                                                              SPOOLDNR  46
```

Subroutine GENMU

```
      SUBROUTINE GENMU(VKTGS,WTML,XMAIN,XNOSE,YCG)                          GENMU
C**** THIS SUBROUTINE PROVIDES THE BRAKING COEFFICIENT OF FRICTION.         GENMU
C****                                                                      GENMU
      DOUBLEPRECISION DTIME,DTIMEJ,TIME                                     CTRL
      PARAMETER (LUIN=3,LUOUT=4)                                            CTRL
      COMMON/CTRL/      DTIME,DTIMEJ,ICOUNT,JDEBUG,KENG,LINENUM,LUMSG,NCOUNT, CTRL
     &                  NEQ,NPAGE,TIME,TIMEROL                              CTRL
      COMMON/RUNWAY/    ABARG,AOAABRK,BRAKMU,BRKFCTR,GAMMARW,GRW,HAGL,      RUNWAY
     &                  HCLEAR,HFLARE,HGEAR,HRUNWAY,IMU,RCR,ROLLMU,TIMEFLD, RUNWAY
     &                  TIMEBRK,TIMEFLP,TIMESBK,TIMESPL,XMU                 RUNWAY
      PARAMETER (ASL=661.48,FPSKTS=1.687806,G= 32.174,                      CONST
     &              RX=57.29577951308,TSLF=59.0,ZERO=0.0)                   CONST
      COMMON/CONST/     ASLSQR5,TWOOVR7                                     CONST
      COMMON/RACURV/    IDBUG,IEXTOR,KURNAM(99)                             RACURV
      COMMON/VALUES/VALIN(15),VFUEL(4),VIDLE(4),VLIFT(2),VAERO(15)          VALUES
      COMMON/TABLES/TFUEL (172),TIDLE (119),TAERO07(72),TAERO08(88),        TABLES
     &                TAERO09(79),TAERO10(62),TAERO11(91),TAERO12(76),      TABLES
     &                TAERO13(64),TAERO14(73),TAERO15(74),TAERO16(66),      TABLES
     &                TAERO17(66),TAERO18(77),TAERO19(73),TAERO20(76),      TABLES
     &                TAERO21(76),TAERO22(81),TAERO23(74)                   TABLES
C****                                                                      GENMU
C**** F-XX CONSTANTS                                                        GENMU
C**** XMAIN  = DISTANCE MAIN GEAR TO CG       LODMAIN = MAIN GEAR LOAD      GENMU
C**** XNOSE  = DISTANCE NOSE GEAR TO CG       LODNOSE= NOSE GEAR LOAD       GENMU
C**** YCG    = HEIGHT OF CG ABOVE GROUND      WTML    = WEIGHT - LIFT       GENMU
      XMAIN  =  9.25                                                        GENMU
      XNOSE  = 54.92                                                        GENMU
      YCG    = 14.00                                                        GENMU
C****                                                                      GENMU
C**** IMU = 0  USE CONSTANT MU MODEL;  IMU = 1  USE MAXMU CURVE             GENMU
C**** IMU = 2  USE GENERIC DRY RUNWAY; IMU = 3  USE GENERIC WET RUNWAY      GENMU
C**** IF IMU = 0, GENMU IS NOT CALLED BY DERIVGR.                           GENMU
      IF    (IMU.EQ.1) THEN                                                 GENMU
C****       RANDOM ACCESS CURVE MAXMU                                       GENMU
C           VALIN(13) = WTML*VKTGS**2/1.0E+09                               GENMU
C           VALIN(14) = RCR                                                 GENMU
C           CALL CURVEL( KURNAM(19),TAERO19,TAERO19,VALIN,VAERO(12),        GENMU
C    &                  IDBUG,IEXTOR,LUMSG)                                 GENMU
C           BRAKMU = VAERO(12)                                              GENMU
C           IF (NCOUNT.EQ.10 .AND.                                          GENMU
C    &         (JDEBUG.EQ.7003.OR.JDEBUG.EQ.7777.OR.JDEBUG.EQ.9999) THEN    GENMU
C              WRITE(LUMSG,7003) VAERO(12)                                  GENMU
C7003          FORMAT(' BRAKMU = ',F10.3)                                   GENMU
C           ENDIF                                                           GENMU
      ELSEIF(IMU.EQ.2) THEN                                                 GENMU
C****       DRY RUNWAY                                                      GENMU
            VKTGSX  = AMAX1((130.0 - VKTGS),ZERO)                           GENMU
            BRAKMU = (0.490 - VKTGSX*0.00127)                               GENMU
```

```
      ELSEIF(IMU.EQ.3) THEN
C****      WET RUNWAY
           IF(VKTGS .LT.  50.0) BRAKMU = (0.3630 - 0.0006737*VKTGS)
           IF(VKTGS .GE.  50.0) BRAKMU = (0.3063 - 0.0022300*VKTGS)
      ENDIF
      BRAKMU  = BRAKMU*BRKFCTR
      XMU     = BRAKMU
      RETURN
      END
```



APPENDIX C: MICROSOFT FORTRAN COMPILER INFORMATION

This section describes the internal software structure used to generate the TOLAND libraries and executables with the Microsoft™ FORTRAN compiler (version 5.1 or later). The location of the subroutines within each library is also detailed.

Executables (.EXE files) are generated from the user provided source code and the following libraries: CURVE.LIB, PANDFQ.LIB, and TOLAND.LIB. The libraries (PANDFQ and CURVE) satisfy external references for UFTAS subroutines such as TABINT, HORP, STASK, and other random access curve file subroutines. The TOLAND.LIB library contains the following object libraries: LIBMAIN.OBJ, LIBTKO.OBJ, LIBLND.OBJ, and LIBGRND.OBJ. These object libraries are generated from their source library counterparts: LIBMAIN.FOR, LIBTKO.FOR, LIBLND.FOR, LIBGRND.FOR, and LIBGENU.FOR, respectively. LIBGENU.OBJ is not included in TOLAND.LIB. Because many of the generic subroutines will be replaced by user specific subroutines, any generic subroutines within the executable file must be explicitly added during the build process. Individual generic subroutine object files are provided for this purpose or a user may generate their own custom general user object library (LIBGENU.OBJ). This object library (along with TOLAND.LIB and user TOLAND source file) would comprise the program list for the executable file build. Table C-1 indicates which subroutines are contained within each library.

TABLE C-1
SUBROUTINE LOCATIONS

LIBMAIN	LIBTKO	LIBLND	LIBGRND	LIBGENU
INITIAL	TAKOFF	LANDNG	ROLL	INICURV
INTX	DERIVAT	STEDYST	DERIVGR	FORCEX
INTG	ERROR	FLARENZ		FXXAERO
HALT		APPROCH		GEFFECT
ATMOSPH		FLARE		FXXENG
SPEED		DERIVAL		SPOOLUP
INTERP		ITRLND		SPOOLDNF
		DGDT		SPOOLDNR
		DVDT		GENMU
		DVTDH		FRETRAC
		DADH		GRETRAC
		DDELTDH		PITCH
		DSIGDH		SPDBRAK
		ZEROX		SPOIL
				TVECTOR

All common blocks for the TOLAND program are contained in separate files with the .CMN extension.

External references satisfied with PANDFQ.LIB, a general purpose flight dynamics library, are found within four sublibraries, LIBAERO, LIBARRAY LIBTRIG, and LIBUTIL. These sublibraries contain aerodynamic, array processing, trigonometric, and miscellaneous subroutines, respectively.

LIBAERO contains UFTAS routines HORP, MACH, QORV, SRHOS, STASK, and VCMACH. LIBARRAY contains UFTAS routines DIVC, IDENT, INVERS, and NORM. LIBTRIG contains routines COSD, SIND, and TAND; they accept arguments in degrees instead of radians. LIBUTIL contains UFTAS routines LBLSORT, RDBFLE, TABINT and WPSDIFF. WPSDIFF is a new name for DUZ2 which performed wild-pointing, smoothing and differentiation of time history data.

Program compilation, input file editing, program execution, output file viewing and output file printing are accomplished with dCOM Directory Commander™ macros. The macro file is named TOLAND.MAC and should be stored in the dCOM subdirectory. The output file printing macro adds a filename, date/time tag to the print spooler and also formats the output to landscape mode.

APPENDIX D: NOMENCLATURE

This section provides an index of program variables. Variables are included here if they are namelist inputs, program outputs, stored within a common block, or passed as a calling argument to a subroutine.

FORTRAN Name	Symbol	Description	Dimension	Usage: COMMON	Usage: SUBROUTINE	Usage: NAME LIST
A	a	Dummy variable passed to subroutine ZEROX as angle of attack argument to function FUNCTN	- - - -		ZEROX	
ABARG	$\bar{a}$	Average deceleration	feet/second^2	*RUNWAY*		
ACCEL	a_t	Acceleration along ground path	feet/second^2	*INTEG*		
ACTION		Action for spoiler or speed brake deflection (e.g. DEPLOY, RETRACT)	- - - -		SPOIL, SPDBRAK	
AFPS	a	Current speed of sound	feet/second	*ATMOS*		
AIRDIST	$S_{g_{TD}}$	Ground distance at touchdown	feet			
AIT	i_t	Thrust incidence angle	degrees	*ENGINE*		
ALPHA	α	Current angle of attack	degrees		APPROCH, DERIVAL, DERIVAT, DERIVGR, DGDT, DVDT, FLARE, FLARENZ, FORCEX, INITIAL, INTX, LANDNG, PITCH, ROLL, STEDYST, TAKOFF	LND
ALPHAJ	α_j	Previous angle of attack from last integration	degrees	*AIRBORN*		
ALPHAX	α	Dummy variable passed to functions DGDT and DVDT	degrees		DGDT, DVDT	
ALPHMX	α_{max}	Maximum angle of attack limit	degrees	*AIRBORN*		
AMACH	M	Mach number	- - - -	*ENGINE*	SPEED	

FORTRAN Name	Symbol	Description	Dimension	Usage: COMMON	Usage: SUBROUTINE	Usage: NAME LIST
AOAABRK	$\alpha_{aerobrk}$	Angle of attack for aerobraking	degrees	*RUNWAY*		ROL2
AOA0FLG	$fl_{\alpha=0}$	Zero angle of attack flag	- - - -	*FLAGS*		
AOA3PT	$\alpha_{3\text{-point}}$	Angle of attack for a 3-point attitude	- - - -	*AIRCRFT*		DATA2
APPDIST	$S_{g_{app}}$	Approach ground distance	feet			
AR	b^2/S_{wing}	Aspect ratio of the wing	- - - -	*AIRCRFT*		
ARRAY		Dummy argument for subroutine ATMOSPH	- - - -		ATMOSPH	
ASL	a_{sl}	Speed of sound at sea level	knots	*CONST*		
ASLSQR5	$a_{sl}\sqrt{5}$	Product of the Speed of sound at sea level and $\sqrt{5}$	knots	*CONST*		
B	b	span of the wing	feet	*AIRCRFT*		
B	B	Dummy variable passed to subroutine ZEROX as angle of attack argument to function FUNCTN	- - - -		ZEROX	
BRAKMU	μ_{brake}	Braking coefficient of *friction*	- - - -	*RUNWAY*		ROL
BRKFCTR		Braking Factor (applied to BRAKMU)	- - - -	*RUNWAY*		ROL2
BRKFLAG	fl_{brake}	Brake flag	- - - -	*FLAGS*		
CD	C_D	Drag coefficient	- - - -		FORCEX	
CGPCT	c_g	Longitudinal center of gravity	percent mean aerodynamic chord	*AIRCRFT*		DATA
CL	C_L	Lift coefficient	- - - -		FORCEX, GEFFECT	
CLALPH	C_{L_α}	Lift curve slope	radians^{-1}	*AIRCRFT*		
CLRHGT	fl_{clrhgt}	Clearance height flag	- - - -	*FLAGS*		
CONFIG		Aircraft Configuration Variable (user definable)	- - - -	*AIRCRFT*		DATA2
CONSTANT		Character value containing either 'VC', VE' or 'MACH' for determining the climb speed derivative with respect to pressure altitude, dV_t/dH	- - - -		DVTDH	
CX	C_X	Force coefficient along flightpath	- - - -	*AERO*		
CY	C_Y	Force coefficient normal to flightpath	- - - -	*AERO*		
DADT	$d\alpha/dt$	Time rate of change of angle of attack	degree/second			

FORTRAN Name	Symbol	Description	Dimension	Usage: *COMMON*	Usage: SUBROUTINE	Usage: NAME LIST
DADTCMD	$(d\alpha/dt)_{cmd}$	Commanded time rate of change of angle of attack	degree/second	*AERO*	PITCH	DATA
DCDGE	$\delta C_{D_{ge}}$	Ground effect delta drag coefficient	- - - -		GEFFECT	
DCDLGR	$\delta C_{D_{gear}}$	Landing gear delta drag coefficient	- - - -		GRETRAC	
DCDX	δC_D	Delta drag coefficient	- - - -	AERO		DATA2
DCLGE	$\delta C_{L_{ge}}$	Ground effect delta lift coefficient	- - - -		GEFFECT	
DCLX	δC_L	Delta lift coefficient	- - - -	*AERO*		DATA2
DDIST	S_i	Incremental ground distance (distance per integration step)	feet		INTG	
DELTA	δ	Current pressure ratio	- - - -	*ATMOS*		
DERIV		Subroutine name variable passed to Runge-Kutta numerical integration subroutine INTX	- - - -		INTX	
DFLAPDT	$\Delta\delta_f/dt$	Rate of change of flap deflection	degree/second		FRETRAC	
DGDTR	a_n	Acceleration normal to flightpath	radian/second^2	*FPINTEG*		
DIST	S_a	Current air distance (ground distance uncorrected for wind)	feet	*INTEG*	INTG	
DISTJ	S_a	Previous air distance (ground distance uncorrected for wind)	feet	*INTEG*	INTG	
DISTMAX	$S_{g_{max}}$	Ground distance limit for takeoff simulation	feet			TKO2
DRIVER		Boundary limit variable on zero finding subroutine	- - - -		ITRLND	
DSBKDT	$\Delta\delta_{sbk}/dt$	Rate of change of speed brake deflection	degree/second		SPDBRAK	
DSPLRDT	$\Delta\delta_{spl}/dt$	Rate of change of spoiler deflection	degree/second		SPOIL	
DTDT	$d\theta/dt$	Current pitch rate	degree/second	*AIRBORN*		
DTDTGEX		Pitch rate capability loss factor in ground effect	- - - -	*AERO*	PITCH	
DTDTMX	$(d\theta/dt)_{max}$	Maximum pitch rate during landing simulation	degree/second	*AIRCRFT*		LND2
DTEMPF	δT_{std}	Delta temperature from standard day	degrees Fahrenheit	*ATMOS*		DATA
DTFAIL	δt_{fail}	Time for the failed engine to lose thrust	seconds	*ENGINE*		DATA2
DTGEAR	δt_{gear}	Time for gear retraction	seconds		GRETRAC	

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	SUBROUTINE	NAME LIST
DTIME	Δt	Runge-Kutta numerical integration step size	seconds	*CTRL*	INTX, PITCH	**DATA2**
DTIMEJ	Δt_j	Runge-Kutta numerical integration step size	seconds	*CTRL*		
DVECTDT	$\Delta\delta_v/dt$	Rate of change of thrust vector angle	degree/second		TVECTOR	
ENGGRP	❖$_{engin}$	Operating Engine Group	- - - -	*CHARV*		**DATA2**
ENGNDX		Engine Spoolup index (user definable)	- - - -		SPOOLUP	
EPR	EPR	Engine pressure ratio	- - - -	*ENGINE*		**DATA2**
ERRFLAG	⚐$_{error}$	Error flag	- - - -	*FLAGS*		
ERROR	ε	Present error for zero finding routine	- - - -		ITRLND	
ERRORJ	ε_j	Previous error for zero finding routine	- - - -		ITRLND	
FACTOR		Multiplicative jump factor on zero finding subroutine	- - - -		ITRLND	
FAILFLG	⚐$_{engfail}$	Engine failure flag	- - - -	*FLAGS*		
FAILGRP	❖$_{fail}$	Fail Engine Group	- - - -	*CHARV*		**DATA2**
FAILMOD		Engine Failure Mode (SEIZE or SPOOL)	- - - -	*CHARV*		**DATA2**
FAILST		Engine Failure State (IDLE or OFF)	- - - -	*CHARV*		**DATA2**
FE	F_E	Ram Drag	pound-force	*ENGINE*		
FG	F_G	Gross Thrust	pound-force	*ENGINE*		
FGPCT	$\%F_G$	Gross Thrust Percentage Increment	percent F_G			**DATA2**
FINDV	⚐$_{findv}$	Steady state condition flag	- - - -		STEDYST, ZEROX	
FLAP	δ_f	Flap deflection	degrees	*AERO*	FRETRAC	**DATA**
FLAPFLG	⚐$_{flap}$	Flap flag	- - - -	*FLAGS*		
FLAP0	δ_{f0}	Initial flap deflection	degrees		INITIAL	
FLPARY	$\#\#_{flap}$	Flap deflection array	degrees	*FLAPDAT*		**TKOARY**
FLPPCT	%Flap	Flap percentage setting	percent max flap	*AERO*		**DATA2**
FLRDIST	$S_{g_{flare}}$	Flare ground distance	feet			
FLT	$\#_{flight}$	Flight number	- - - -	*AIRCRFT*		**DATA2**
FLTNDX		Flight index (user definable)	- - - -			**DATA2**
FPACCEL	a_t	Acceleration along flightpath	feet/second^2	*FPINTEG*		

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	SUBROUTINE	NAME LIST
FPCTFLG	$\text{Fb}_{\%flap}$	Flap percentage flag	- - - -	*FLAGS*		
FPDIST	$S_{a_{fp}}$	Flightpath air distance	feet	*FPINTEG*		
FPSKTS		Conversion factor from knots to feet per second	feet/second/knot	*CONST*		
FPVTAS	V_t	Flightpath true airspeed	feet/second	*FPINTEG*	DGDT, DVDT, STEDYST	
FUNCTN		Function DGDT or DVDT passed to Function ZEROX zero finding routine	- - - -		ZEROX	
G	g	Acceleration due to gravity	feet/second^2	*CONST*		
GAMMA	γ	Flightpath angle (inertial reference frame)	degrees			
GAMMAPP	$\gamma_{approach}$	Approach flightpath angle (inertial reference frame)	degrees	*AIRBORN*	APPROCH, FLARENZ	LND
GAMMAR	γ	Flightpath angle	radians	*FPINTEG*	DGDT, DVDT, INTG, SPEED, STEDYST	
GAMMARW	γ_{runway}	Runway slope	degrees	*RUNWAY*		DATA
GAMMATD	$\gamma_{touchdown}$	Touchdown flightpath angle (inertial reference frame)	degrees		FLARENZ	
GDIST	S_g	Ground distance	feet		APPROCH, FLARE, INTG, ROLL	
GEFLAG	$\text{Fb}_{geffect}$	Ground effect flag	- - - -	*FLAGS*		
GROUP		Group of variables to be initialized	- - - -		INITIAL	
GRW	γ_{runway}	Runway slope	radians	*RUNWAY*		
GWT	W_{gross}	Gross weight	pound-force	*AIRCRFT*	DGDT, DVDT, INTG	
GWT0	W_{gross0}	Initial gross weight	pound-force		INITIAL	DATA
HAGL	H_{agl}	Current altitude above liftoff point for takeoffs	feet	*RUNWAY*	APPROCH, FLARENZ, GEFFECT, GRETRAC	
HAGL	H_{agl}	Current altitude above runway for landings	feet	*RUNWAY*	APPROCH, FLARENZ, GEFFECT, GRETRAC	

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	Usage SUBROUTINE	Usage NAME LIST
HC	H_c	Pressure altitude	feet		DADH, DDELTDH, DSIGDH	
HCLEAR	H_{clear}	Obstacle clearance height	feet	RUNWAY	APPROCH, FLARENZ	**DATA**
HCLIMB	H_{climb}	Height simulation switches to constant airspeed climb	feet			**TKO2**
HCLMOUT	$H_{climbout}$	Height simulation switches to constant theta climb	feet			**TKO2**
HFLARE	H_{flare}	Flare initiation height	feet	*RUNWAY*	APPROCH, FLARENZ	**LND**
HGEAR	H_{gear}	Landing Gear Retraction Height	feet	*RUNWAY*		**TKO**
HMAX	H_{max}	Termination Height Limit	feet			**TKO**
HRUNWAY	H_{runway}	Runway pressure altitude	feet	*RUNWAY*	APPROCH, FLARENZ	**DATA**
HVCTARY	$\#\#_{hvect}$	Vectored thrust altitude array	feet	*VECTDAT*		**TKOARY**
HVECT	H_{vect}	Current vectored thrust altitude array element	feet		TVECTOR	
HZ	H_z	Height of wing above the bottom of the tires	feet	*AIRCRFT*		
ICOUNT	i_{count}	Integration loop initializing variable	- - - -	*CTRL*		
IDBUG		UFTAS debug code	- - - -	*RACURV*		**DATA2**
IEXTOR		Extrapolation override code	- - - -	*RACURV*		
IFLAP	i_{flap}	Current element of arrays FLPARY and VFLPARY	- - - -	*FLAPDAT*		
IGEAR	i_{gear}	Current element of array LGRARY	- - - -	*GEARDAT*		
IMU		Braking coefficient selector	- - - -	*RUNWAY*		**ROL**
IVECT	i_{vect}	Current element of arrays XNUARY, HVCTARY, and VVCTARY	- - - -	*VECTDAT*		
JDEBUG		TOLAND Debug Code	- - - -	*CTRL*		**DATA2**
JFLAG	⚑$_J$	Subroutine ITRLND search status variable	- - - -		ITRLND	
KENG		Subroutine 'FXXENG' output switch	- - - -	*CTRL*		
KURNAM	$\#\#_{crvname}$	Random access curve file name array	- - - -	*RACURV*		
LGRARY	$\#\#_{gear}$	Landing gear drag array	- - - -	*GEARDAT*		
LGRFLAG	⚑$_{gear}$	Landing gear flag	- - - -	*FLAGS*		
LIFTOFF	⚑$_{liftoff}$	Liftoff flag	- - - -	*FLAGS*		

FORTRAN Name	Symbol	Description	Dimension	Usage: *COMMON*	Usage: SUBROUTINE	Usage: NAME LIST
LINENUM		Current line number of current page of output	- - - -	*CTRL*		
LOADING		Aircraft External Stores Loading Variable (user definable)	- - - -	*AIRCRFT*		DATA2
LUCURV	🗀$_{curve}$	Logical unit for the random access curve file	- - - -	*CINDEX*		
LUIN	🗀$_{in}$	Logical unit for program input	- - - -	*CTRL*	HALT	
LUMSG	🗀$_{msg}$	Logical unit for message output	- - - -	*CTRL*	HALT	DATA2
LUOUT	🗀$_{out}$	Logical unit for program output	- - - -	*CTRL*	ERROR, HALT PITCH	
MANUVR		Maneuver input to main program	- - - -	*CHARV*	PITCH	
MANUVR		Maneuver passed to subroutine PITCH	- - - -		PITCH	
MAXSIZF		Maximum size of arrays FLPARY and VFLPARY	- - - -	*FLAPDAT*		
MAXSIZG		Maximum size of array LGRARY	- - - -	*GEARDAT*		
MAXSIZV		Maximum size of arrays XNUARY, HVCTARY, and VVCTARY	- - - -	*VECTDAT*		
MSTRNDX	###$_{master}$	Master index array for random access curve file lookups	- - - -	*CINDEX*		
MVR		First three characters of MANUVR	- - - -	*CHARV*		
NCOUNT	n_{count}	Integration loop counter		*CTRL*		
NDXLEN		Length of master index array, MSTRNDX	- - - -	*CINDEX*		
NENG	n_{eng}	Total number of engines	- - - -	*ENGINE*		
NEQ	n_{eq}	Number of equations of motion to be integrated	- - - -	*CTRL*	INTX	
NPAGE	n_{page}	Number of lines of output per page	- - - -	*CTRL*		
ODIST	S_{obs}	Pre-Flare ground distance	feet		FLARE	
OVERFLG	⚐$_{over}$	Flare height over obstacle clearance height flag	- - - -	*FLAGS*		
PLA	$\angle_{pla}$	Average power lever angle	degrees		FXXENG	
PLAARY	###$_{pla}$	Power lever angle array	degrees		FXXENG	
PRESALT	h_p	Pressure altitude	feet	*FPINTEG*	ATMOSPH	

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	SUBROUTINE	NAME LIST
PRESS	p_a	Current ambient pressure	pound-force/feet^2	*ATMOS*		
PWRCODE		Aircraft power code	- - - -		FXXENG	**DATA2**
QS	qS	Product of dynamic pressure and reference wing area	pound-force	*AERO*	SPEED	
RC		Engine thrust rating code	- - - -	*ENGINE*		**DATA2**
RCR		Runway condition reading	- - - -	*RUNWAY*		**ROL**
REVFLAG	$⚑_{rev}$	Reverse thrust flag	- - - -	*FLAGS*		**DATA2**
REVNDX		Reverse engine spoolup index (user definable)	- - - -	*ENGINE*		**DATA2**
REVRSE	$⚑_{revrse}$	Reverse thrust active flag	- - - -	*FLAGS*		
RHO	ρ	Current density	slugs/feet^3	*ATMOS*		
RKAIR	$\#_{RKair}$	Runge-Kutta flightpath array	- - - -	*FPINTEG*		
RKGRND	$\#_{RKgrnd}$	Runge-Kutta ground roll array	- - - -	*INTEG*		
ROC	R/C	Current rate of climb	feet/second	*FPINTEG*		
ROCFPM	R/C	Current rate of climb	feet/minute	*AIRBORN*	ERROR	
ROCTD	R/C_{TD}	Rate of climb at touchdown	feet/second		FLARE	
ROLLMAX	$(t_{roll})_{max}$	Takeoff Ground Roll Time Limit	seconds			**TKO2**
ROLLMU	μ_{roll}	Rolling coefficient of friction	- - - -	*RUNWAY*		**DATA2**
ROTATE	$⚑_{rotate}$	Rotation flag	- - - -	*FLAGS*		
RTOFLAG	$⚑_{rto}$	Refused takeoff flag	- - - -	*FLAGS*		
RX		Conversion factor from radians to degrees	degree/radian	*CONST*		
SBKEND	$\delta_{sbk_{end}}$	Final spoiler deflection	degrees		SPDBRAK	
SBKFLAG	$⚑_{speedbrake}$	Speed Brake flag	- - - -	*FLAGS*		
SIGMA	σ	Current density ratio	- - - -	*ATMOS*		
SINKTD		Sink rate at touchdown	feet/second		FLARENZ	**LND**
SPDBRK	δ_{sbk}	Speed Brake deflection	degrees	*AERO*	SPDBRAK	
SPDBRK0	δ_{sbk0}	Initial Speed Brake deflection	degrees		INITIAL, SPDBRAK	**DATA2**
SPLFLAG	$⚑_{spoiler}$	Spoiler flag	- - - -	*FLAGS*		**LND2**

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	SUBROUTINE	NAME LIST
SPLREND	$\delta_{spl_{end}}$	Final spoiler deflection	degrees		SPOIL	
SPOILER	δ_{spl}	Spoiler deflection	degrees	*AERO*	SPOIL	
SPOOL	⚐$_{spool}$	Engine spooling status flag	- - - -	*FLAGS*	SPOOLDNF, SPOOLDNR, SPOOLUP	
STEADY	⚐$_{steady}$	Steady state flag	- - - -	*FLAGS*		
SWING	S_{wing}	Reference wing area	feet^2	*AIRCRFT*		
T	[illegible]$_{RK}$	Runge-Kutta numerical integration array passed to subroutine INTX T is either the values of common block INTEG or FPINTEG.	- - - -		INTX	
TEMPR	T	Current temperature	degrees Rankine	*ATMOS*		
TERMFLG	⚐$_{term}$	Program terminate flag	- - - -	*FLAGS*		
TERMMSG	[illegible]$_{term}$	Termination message	- - - -	HALT		
THETA	θ	Current temperature ratio	- - - -	*ATMOS*		
THETAF	θ	Current pitch attitude	degrees	*AIRBORN*		
THRCRV		Thrust curve type (user defined)	- - - -	*CHARV*		**DATA2**
THRUST	F_N	Net Thrust	pound-force	*ENGINE*	FLARENZ	
THTCLM	θ_{climb}	Pitch attitude tracked between liftoff and H_{climb}	degrees			**TKO**
THTFLY	θ_{fly}	Pitch attitude tracked above H_{clmout}	degrees			**TKO2**
THTMAX	ϑ_{max}	Current maximum pitch attitude limit	degrees	*AIRCRFT*		
THTROT	θ_{rotate}	Pitch attitude tracked between rotation and liftoff	degrees			**TKO**
THTTOL	$\theta_{tolerance}$	Pitch attitude tolerance	degrees			**TKO2**
TIME	t	Elapsed time	seconds	*CTRL*	INTX, SPOOLDNF, SPOOLDNR, SPOOLUP	
TIMEBRK	t_{brake}	Time delay between touchdown and brake application for a landing or the time delay between engine failure and brake application for a refused takeoff.	seconds	*RUNWAY*		**ROL2**

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	Usage SUBROUTINE	NAME LIST
TIMEFLD	t_{failed}	Elapsed time from engine failure	seconds	*RUNWAY*		
TIMEFLP	t_{flap}	Time delay between touchdown and flap retraction for a landing or the time delay between engine failure and flap retraction for a refused takeoff.	seconds	*RUNWAY*		**ROL2**
TIMEIDL	t_{idle}	Time delay between engine failure and engine spool down initiation for a refused takeoff.	seconds	*RUNWAY*		**ROL2**
TIMEMAX	t_{max}	Time limit for takeoff simulation	seconds			**TKO2**
TIMEREV	$t_{reverse}$	Time delay between touchdown and engine spool up initiation of thrust reversers for a landing or the time delay between engine failure and spool up initiation of thrust reversers for a refused takeoff.	seconds	*RUNWAY*		**ROL2**
TIMEROL	t_{roll}	Elapsed time of the ground roll	seconds	*CTRL*		
TIMESBK	$t_{speedbrake}$	Time delay between touchdown and speedbrake deployment for a landing or the time delay between engine failure and speedbrake deployment for a refused takeoff.	seconds	*RUNWAY*		**ROL2**
TIMESPL	$t_{spoiler}$	Time delay between touchdown and spoiler deployment for a landing or the time delay between engine failure and spoiler deployment for a refused takeoff.	seconds	*RUNWAY*		**ROL2**
TKOTYPE		Takeoff type (either STATIC or ROLLING)	- - - -	*CHARV*		**TKO2**
TOLRNCE	ε_t	Maximum tolerance constant for zero finding routines	- - - -		ITRLND, ZEROX	
TSLE	T_{sl}	Standard temperature at sea level	degrees Fahrenheit	*CONST*		
TWOOVR7	2/7	Ratio of two over seven	- - - -	*CONST*		

FORTRAN Name	Symbol	Description	Dimension	Usage COMMON	SUBROUTINE	NAME LIST
VAR2	var_2	Dummy variable passed to subroutine ZEROX as an argument to FUNCTN	- - - -	ZEROX		
VAR3	var_3	Dummy variable passed to subroutine ZEROX as an argument to FUNCTN	- - - -	ZEROX		
VAR4	var_4	Dummy variable passed to subroutine ZEROX as an argument to FUNCTN	- - - -	ZEROX		
VCLMOUT	V_{clmout}	Airspeed tracked between H_{climb} and H_{clmout}				TKO2
VECTFLG	Fb_{vect}	Thrust vectoring flag	- - - -	*FLAGS*		
VFFLAG	Fb_{vf}	Engine failure velocity flag	- - - -	*FLAGS*		
VFLPARY	$\#\#_{Vflap}$	Flap deflection airspeed array	knots calibrated airspeed	*FLAPDAT*		TKOARY
VISCOSK	v	Current kinematic viscosity	$feet^2$/second	*ATMOS*		
VHAS	V_h	Current horizontal airspeed	feet/second	*FPINTEG*		
VKAPRK	$V_{aer\ brk}$	Aerobraking termination airspeed	knots calibrated airspeed	*AIRSPED*		ROL2
VKAPP	$V_{approach}$	Approach airspeed for landing simulation	knots calibrated airspeed	*AIRSPED*	FLARENZ	LND
VKBRAKE	V_{brake}	Wheel braking airspeed	knots calibrated airspeed	*AIRSPED*		ROL2
VKCAS	V_c	Current calibrated airspeed	knots	*AERO*	SPEED	
VKEAS	V_e	Current equivalent airspeed	knots		SPEED	
VKEND	V_{end}	Simulation end airspeed	knots calibrated airspeed	*AIRSPED*		TKO2
VKFAIL	V_{fail}	Engine failure airspeed	knots calibrated airspeed	*AIRSPED*		TKO
VKFLAP	V_{flap}	Flap retraction airspeed	knots calibrated airspeed	*AIRSPED*	FRETRAC	TKO2
VKFLPMX	$(V_{flap})_{max}$	Maximum flap retraction airspeed	knots calibrated airspeed	*AIRSPED*		
VKMCG	V_{mcg}	Minimum control airspeed on the ground	knots calibrated airspeed	*AIRSPED*		DATA2
VKROTAT	V_{rotate}	Rotation airspeed	knots calibrated airspeed	*AIRSPED*		TKO
VKSTART	V_{start}	Simulation start groundspeed	knots calibrated airspeed	*AIRSPED*		TKO2
VKTAS	V_t	Current true airspeed	knots	*AERO*	SPEED	

FORTRAN Name	Symbol	Description	Dimension	Usage: *COMMON*	Usage: SUBROUTINE	Usage: NAME LIST
VKTGS	V_g	Current groundspeed	knots		FLARE, GENMU, SPEED	
VKTGTD	$(V_g)_{TD}$	Groundspeed at touchdown	knots			
VKWIND	V_{wind}	Wind speed	knots	*AIRSPED*		DATA
VTANGLE	v	Vectored thrust angle	degrees	*ENGINE*	TVECTOR	
VTAS	V_t	Current true airspeed	feet/second	*INTEG*	DVTDH	
VTASJ	V_{t_j}	Previous true airspeed	feet/second	*INTEG*		
VTASX	V_{t_x}	Dummy argument true airspeed passed to subroutine SPEED	feet/second		SPEED	
VTGS	V_{t_g}	Current true groundspeed	feet/second		SPEED	
VVCTARY	$\#\#_{vvect}$	Vectored thrust airspeed array	knots calibrated airspeed	*VECTDAT*		TKOARY
VVECT	V_{vect}	Current vectored thrust airspeed array element	knots calibrated airspeed		TVECTOR	
VWIND	V_{wind}	Wind speed	feet/second	*AIRSPED*	INTG, SPEED	
WFUEL	W_f	Fuel flow	pound-meter/hour	*ENGINE*		
WNGLOD	W/S_{wing}	Wing loading at brake release	pound-force/feet^2	*AIRCRFT*		
WRITITR	$Fb_{writitr}$	Write iterations flag for subroutine FLARE	- - - -	*FLAGS*		LND2
WTML	W_{-lift}	Gross weight minus lift	pound-force		GENMU	
XANS	x_{interp}	X-axis coordinate for interpolation to calculate INTERP function value	- - - -		INTERP	
XENG	X_{eng}	Current engine multiplicative factor	- - - -	*ENGINE*		DATA2
XENGEND	X_{engend}	Engine multiplicative factor at the end of a throttle transient	- - - -		SPOOLDNF,SPOOLDNR	
XENGFLD	X_{failed}	Failed engines multiplicative factor	- - - -	*ENGINE*		DATA2
XENGOUT	X_{engout}	Engine Multiplicative Factor after engine failure	- - - -	*ENGINE*		
XENGTRN	$X_{transient}$	Engine Multiplicative Factor of transient engines	- - - -		SPOOLDNF, SPOOLDNR, SPOOLUP	

FORTRAN Name	Symbol	Description	Dimension	Usage		
				COMMON	SUBROUTINE	NAME LIST
XENG0	X_{eng0}	Engine multiplicative factor at the beginning of a throttle transient	- - - -		SPOOLUP	
XIDLE	X_{idle}	Idle Engine Multiplicative Factor for a single engine	- - - -	*ENGINE*		
XLF	n_z	Current normal load factor	gravity	*AIRBORN*		
XLFJ	n_{z_j}	Previous normal load factor from last integration	gravity	*AIRBORN*		
XLFLARE	$(n_z)_{flare}$	Flare normal load factor limit	gravity		FLARENZ	
XLFMAX	$(n_z)_{max}$	Maximum normal load factor limit	gravity	*AIRCRFT*	FLARENZ	
XMAIN	x_{main}	Distance from main gear to cg	feet		GENMU	
XMIL	X_{mil}	Military Thrust Engine Multiplicative Factor for a single engine	- - - -	*ENGINE*		
XMU	μ	Current coefficient of friction	- - - -	*RUNWAY*		
XNOSE	x_{nose}	Distance from nose gear to cg	feet		GENMU	
XNOW	x	Current x-axis variable for interpolation boundary	- - - -		INTERP	
XNUARY	ϕ_v	Vectored thrust angle array	degrees	*VECTDAT*		TKOARY
XPAST	x_j	Previous x-axis variable for interpolation boundary	- - - -		INTERP	
YCG	y_{cg}	Height of cg above ground	feet		GENMU	
YNOW	y	Current y-axis variable for interpolation boundary	- - - -		INTERP	
YPAST	y_j	Previous y-axis variable for interpolation boundary	- - - -		INTERP	
ZERO		Constant = 0.0	- - - -	*CONST*		
ZFN	Z_{FN}	User definable variable passed through common ENGINE	- - - -	*ENGINE*		
	$\Uparrow$	Page Eject Symbol in FORTRAN Source Listing*	- - - -			

* The page eject symbol is an unprintable ASCII character of value 014 decimal. It appears as the symbol for the planet Venus in the actual source code within several format statements. This symbol is substituted so actual page ejects will not occur when the source code is printed.

This page intentionally left blank.

APPENDIX E: INDEX

—V—

—W—

—X—

—Y—

—Z—